26811

National Library of Canada | Bibliothèque nationale du Canada

CANADIAN THESES ON MICROFICHE | THÈSES CANADIENNES SUR MICROFICHE

NAME OF AUTHOR/NOM DE L'AUTEUR DAVID KUI KWONG LAM

TITLE OF THESIS/TITRE DE LA THÈSE Fast Inversion of Large Raw-wise Kronecker Matrices with an Application To Optical Inverse Scattering Problem

UNIVERSITY/UNIVERSITÉ UNIVERSITY OF ALBERTA

DEGREE FOR WHICH THESIS WAS PRESENTED/GRADE POUR LEQUEL CETTE THÈSE FUT PRÉSENTÉE PH. D.

YEAR THIS DEGREE CONFERRED/ANNÉE D'OBTENTION DE CE GRADE 1975

NAME OF SUPERVISOR/NOM DU DIRECTEUR DE THÈSE DR. A. Wouk & DR. H.G. Schmidt-Weinmar

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

DATED/DATÉ Sept 9, 1975 SIGNED/SIGNÉ David Lam

PERMANENT ADDRESS/RÉSIDENCE FIXE 5711, 115 STREET,
EDMONTON, ALBERTA
CANADA T6H 3P5

NL-91 (3-74)

THE UNIVERSITY OF ALBERTA

FAST INVERSION OF LARGE ROW-WISE KRONECKER MATRICES WITH AN APPLICATION TO OPTICAL INVERSE SCATTERING PROBLEM

by

(C) DAVID KUI KWONG LAM

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

IN

COMPUTATIONAL OPTICS

DEPARTMENT OF COMPUTING SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL, 1975

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled FAST INVERSION OF LARGE ROW-WISE KRONECKER MATRICES WITH AN APPLICATION TO INVERSE SCATTERING PROBLEM, submitted by David Kui Kwong Lam in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Arthur Wouk
Co-supervisor

Co-supervisor

C. R. Jones

W. A. Davis

Henry C. Andrews
External Examiner

Date: September 9, 1975

ABSTRACT

The properties of eleven types of matrix sequences are studied towards the development of fast matrix inversion algorithms. Among those sequences are sequences of special Vandermonde matrices, row-wise Kronecker matrices, composite matrices, etc.

An m-plex Kronecker system of equations is studied. A fast solution of the m-plex system turns out to be a usual multiplication of a rectangularized constant vector with Kronecker factor matrices inverted and transposed individually.

This fast solution of a system of equations, together with the fast inversion of row-wise Kronecker matrices and special Vandermonde matrices, provides a technique of economically solving a system of thousands of equations arising with the optical inverse scattering problem.

ACKNOWLEDGEMENT

I am indebted to Dr. A. Wouk and Dr. H.G. Schmidt-Weinmar, both being my co-supervisors, for their very kind guidance. Dr. Wouk gave useful advice all the way through the course of research. His suggestion of incorporating the inversion techniques for Vandermonde matrices led to a major breakthrough in the most difficult part of my project. Dr. Schmidt-Weinmar initiated the research topic and guided my work along the line of his optical research, which was supported by the grant from National Research Council of Canada.

I wish to express my appreciation of the advice, criticism, and help from Dr. H.C. Andrews, Dr. S. Cabay, Dr. W.A. Davis, Dr. C.R. James, Dr. J.R. McGregor, and Dr. L. Schubert, all of whom have contributed so much towards the perfection of the thesis.

I am also grateful to Mrs. Mary Yiu for typing up the manuscript.

The financial assistance provided by the Department of Computing Science was valuable for carrying out my research at the University.

CONTENTS

	Page
I. INTRODUCTION	1
1.1 Inverse Scattering Problem	1
1.2 Conventional Methods of Solving a Large System of Equations	5
1.3 Redundancy Techniques in Matrix Decomposition	8
1.4 Solution of Inverse Scattering Problem by Matrix Decomposition	9
PART ONE FAST INVERSION OF MATRICES	13
II. SELF-INVERTING SEQUENCES OF MATRICES	14
2.1 Matrix Sequence	14
2.2 I' Sequence	17
2.3 A Sequence	19
III. QUASI-UNITARY SEQUENCES	23
3.1 Quasi-unitary Sequences	23
3.2 Diagonal Sequence	24
3.3 B Sequence	25
IV. VANDERMONDE SEQUENCES	29
4.1 Vandermonde Sequence	29
4.2 C Sequence	35
4.3 E Sequence	38
4.4 Fourier Transform Sequence	41

		Page
V.	COMPOSITE SEQUENCES	44
5.1	H Sequence	44
5.2	Kronecker Sequence	47
5.3	Row-wise Kronecker Sequence	50
VI.	SYSTEM OF EQUATIONS WITH KRONECKER COEFFICIENT MATRICES	60
6.1	Solution of System by Matrix Inversion	60
6.2	Solution of a 2-plex System by Rectangularizing the Unknown Vector	61
6.3	Solution of an m-plex System by Rectangularizing the Unknown Vector	69
6.4	Matrix Inversion by Rectangularization Techniques	71
6.5	Summary	72
PART TWO	SOLUTION TO INVERSE SCATTERING PROBLEM	74
VII.	OUR APPROACH TO A COMPUTATIONAL SOLUTION	75
7.1	The Physical Problem	75
7.2	Discrete Representations	81
7.3	The System of Equations	85
VIII.	METHOD OF SOLUTION	88
8.1	Outline of the Method	88
8.2	Creation of Row Redundancies	89
8.3	Creation of Column Redundancies	91
8.4	Algebraic Solution	92
8.5	Computing Time	97
8.6	Storage Space	98

		Page
IX.	SELECTION OF PHASE ANGLES	100
9.1	Principles of Selection	100
9.2	Graphical Representation of Phase Angles	102
9.3	Creation of Column Redundancies by Selecting Phase Angles (the First J_3 Rows of P)	105
9.4	Creation of Column Redundancies by Selecting Phase Angles (the Remaining Rows of P)	115
9.5	Stabilizing the Vandermonde Matrices	116
9.6	Factor Matrices belonging to the E Sequence	122
X.	A COMPUTED EXAMPLE	126
10.1	A Simplified Physical System	126
10.2	Selection of Phase Angles	126
10.3	Computer Programs	132
XI.	VARIATION OF OUR METHOD OF SOLUTION	133
11.1	Factor Matrices belonging to the Fourier Transform Sequence	133
PART THREE	ERROR ANALYSIS	136
XII.	CONDITION NUMBERS AND ERROR BOUNDS	137
12.1	Turing's N-Condition Number	137
12.2	Sequence E	139
12.3	Fourier Transform Sequence	140
12.4	Sequence H	141
12.5	Kronecker Sequence	141

		Page
XII.	(cont'd)	
12.6	Vandermonde Sequences	142
12.7	Error Bound of the Presented Method of Solution	144
12.8	Summary	146
XIII.	DISCUSSION	148
13.1	Lensless Reconstruction	148
13.2	Reconstruction of Objects hear the Resolution Limit	150
13.3	Fast Inversion of Large Matrices	151
13.4	The Method of 3-D Reconstruction	153
13.5	Overall Result of the Method Presented	156
13.6	Proposed Directions for Future Research	157
BIBLIOGRAPHY		158
APPENDICES		
Appendix I	Kronecker Product of Matrices	163
Appendix II	The Values of Phase Angles Selected by Computer	165
Appendix III	Computer Programs to Solve for Scattered Near-field	167
Appendix IV	Digitization of Scattering Formula	196
Appendix V	Propagation of Light in a Quasi-Homogeneous Medium	201
Appendix VI	Solution of Non-Homogeneous Helmholtz Partial Differential Equation	205

I. INTRODUCTION

1.1 Optical Inverse Scattering Problem

A coherent laser beam passes through a semi-transparent, weakly scattering object and forms an interference pattern under far-field conditions. Is it possible to determine the optical structure of such an object making use of the information recorded? This is the optical inverse scattering problem.

An interference pattern records only the magnitude of an optical wave. The phase of such a wave cannot be recorded directly by any instruments of today's technology. Without this information, the spatial distribution of the phase of a propagated wavefront, and hence the optical structure of a scatterer, cannot be reconstructed. For this reason, the optical inverse scattering problem has been considered a difficult if not an unsolvable one.

A breakthrough was made by Gabor (1951) who reconstructed the image of a three-dimensional reflecting object from a two-dimensional photographic recording. His technique was later developed to become holography, and convinced people that some phase information was indeed recorded in an interference pattern. Contributions due to Goodman (1967), Lesem, Hirsch, Jordan (1967, 1968), Hickling (1968) proved that a three-

dimensional image can be reconstructed not only by optical methods but also by computers from holographic data. Wolf (1970, 1971) and Carter (1970) studied three-dimensional transmitting objects by off-axis holography. Wolf concluded that some of the spatial frequencies of such an object can be detected from holographic data but that, in order to achieve a complete three-dimensional reconstruction, the object under study has to be rotated. He pointed out that holographic reconstruction may be good to a resolution of about 9 times the vacuum wavelength of light.

Another approach towards the inverse scattering problem parallel to holography went on in x-ray crystallography. X-ray radiation has a much shorter wavelength than an optical laser. Kendrew (1962) and Perutz (1964) attached a known molecule to the molecule of unknown structure, to measure both magnitude and phase of the diffracted x-ray radiation, and were able to sketch the profile of the object molecular structure, by comparing the interference patterns obtained with and without the known molecule attached to the object molecule. Sayre (1974) claimed that computer reconstruction from x-ray patterns carried out by IBM researchers achieved a spatial resolution of 1.5 Å. Both holography and x-ray crystallography represent a diffracted field by uniform

plane waves and process both the magnitude and the phase transmitted by these waves.

There is a recent attempt to measure and to process the magnitude and the phase of plane-wave components of the nonuniform type of the field arising from a diffracting or refracting object. This new approach may be divided into three steps:

A. To recover from interference fringe patterns the spatial distribution of magnitude and phase of an optical-wave field in the far-field region.

B. To compute the spatial distribution of the object near field from the complex-amplitude field data in the far-field region.

C. To determine the optical structure of a scattering object from the near-field data.

Step A was solved by Schmidt-Weinmar (1973(1), 1975(1)), who proposed a three-reference-beam method. By this method, the magnitude and the phase at any point in the far-field region can be computed from three interference fringe patterns, obtained with one and the same object field superimposed upon three versions of a coherent reference field. Schmidt-Weinmar (1975(2)) also published a theory concerning step B. Step C has not yet been completed, but preliminary results concerning step C were presented by Schmidt-Weinmar (1971,

1973(2)) to the Optical Society of America.

With the new approach, the scattered field at any point in the far-field region is considered to be the global effect of many Huygen's point source fields, that emerge from inside the scattering object. These source fields, being activated by the incident laser beam, may number in the thousands. In mathematical terms, the spatial distribution of the scattered far field is related to the spatial distribution of these point sources by a linear mapping. To determine the spatial distribution of these point sources requires the solution of a linear system of possibly thousands of equations. This system of equations is neither sparse nor diagonally dominant. Solving it by Gaussian Elimination is impracticable owing to the hundreds of pages of storage space and the amount of CPU time required. Hence, a special computing technique is sought that requires less storage and computing time than any of the conventional methods for the solution of such a system of equations.

1.2 Conventional Methods of Solving a Large System of Equations

1.2a Solving a System by Matrix Inversion

Consider a linear algebraic system over the complex field

$$M X = Y \qquad (1.2.1)$$

where M is a non-singular square matrix of size (n,n) and Y is a matrix of size (n,m) containing m sets of data. The system will yield

$$X = M^{-1} Y \qquad (1.2.2)$$

where X has a dimension (n,m) giving the m sets of unique solutions to (1.2.1). If M is unitary, (1.2.2) will simply become

$$X = M^{H} Y \qquad (1.2.3)$$

M^{H} being the Hermitian transpose of matrix M. Otherwise, M^{-1} needs to be evaluated and the lengthy calculations normally require the use of an electronic computer.

Apart from obtaining the solution X by inverting the coefficient matrix M, (1.2.1) can be solved by other methods as well. These methods usually fall into two categories: direct methods and iterative methods. Each method has its advantages and disadvantages and selection of these methods depends largely upon the structure of the coefficient matrix M.

1.2b Direct Methods

The major difficulty in the use of direct methods lies in the number of arithmetic operations required. Table 1 shows the number of operations needed by various methods, for the case in which Y is an (n,1) vector, as summarized by Westlake (1968). With the exception of triangular and tridiagonal systems, all direct methods need a number of multiplications proportional to n^3. When n is large, say, n is larger than 20, direct methods will be costly. The problem will become particularly serious when M contains many elements whose absolute values are less than one. Repeated multiplications of these small elements may result in underflow.

Another weak point of the direct methods is that they usually involve too many subtractions, which will easily magnify percentage errors. Normally direct methods are considered applicable to small systems of equations. For large systems, direct methods are used only when no other better methods can be found.

1.2c Iterative Methods

One of the intrinsic advantages of these methods is that errors due to computer round-off etc. may be damped out as the iteration continues. The major disadvantage of all iterative methods is that convergence usually requires a diagonally dominant coefficient

TABLE 1
Number of arithmetic operations
(A is an $n \times n$ matrix)

Method	$AX = b$		
Direct	÷	×	+ & −
Gaussian Elimination	n	$\frac{1}{3}n^3 + n^2 - \frac{1}{3}n$	$\frac{1}{3}n^3 + n^2 - \frac{4}{3}n$
Jordan	n	$\frac{1}{2}n^3 + n^2 - \frac{1}{2}n$	$\frac{1}{2}n^3 - \frac{1}{2}n$
Doolittle	n	$\frac{1}{3}n^3 + n^2 - \frac{1}{3}n$	$\frac{1}{3}n^3 + n^2 - \frac{4}{3}n$
Cholesky (symmetric)	n root recip.	$\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n$	$\frac{1}{6}n^3 + n^2 - \frac{7}{6}n$
Crout		$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$	
Triangular system	n	$\frac{1}{2}n^2 - \frac{1}{2}n$	$\frac{1}{2}n^2 - \frac{1}{2}n$
Product form of the inverse		$\frac{1}{2}n^3 + \frac{1}{2}n^2$	
Tridiagonal	$2n$	$4(n-1)$	$4(n-1)$
Iterative	**Per Iteration**		
	÷	×	+ & −
Jacobi and Seidel	n	n^2	$n^2 - n$
Steepest descent (gradient)	1	$2n^2 + 3n$	$2n^2 + 2n - 2$
Conjugate gradient (not symm.)	1	$3n^2 + 6n + 2$	$3n^2 + 3n - 3$
Successive overrelaxation	n	$n(n+1)$	n^2 $5n$
Peaceman–Rachford	n	$2n^2 + 8n - 8$	$2n^2 + 8n - 8$
Newton–Raphson Analogue	0	$2n^2$	$2n^2 - 2n + n$

matrix. For a suitable class of systems of equations, iterative methods are very efficient, while for other classes, iterative methods need not even converge.

When Y is an (n,m) matrix, both direct methods and iterative methods have to be applied m times in order to get the m sets of answers for x. Therefore, the procedure of inverting the coefficient matrix preliminary to solving the system of equations is often favoured, since we only need to invert M once for all m sets of data.

1.3 Redundancy Techniques in Matrix Decomposition

Good (1958) showed that we may encounter matrices that can be decomposed into a product of several sparse matrices, and that in this case manipulating the sparse matrices can save a number of mathematical operations and storage space. He suggested the technique can be applied to speed up matrix-vector multiplication.

Cooley and Tukey (1965) deliberately created redundant factors in a Fourier transform matrix so that the matrix can be decomposed into two or more sparse matrices for easy manipulation. This is the well-known Fast Fourier Transform algorithm, which paved the way for development of the Fast Hadamard Transform, etc., at a later date. Theilheimer (1969) formulates the actual

procedure for explicitly carrying out such a matrix decomposition.

Andrews and Kane (1970) applied the techniques to build a Kronecker matrix model, and they proved that many fast transform matrices are in fact special cases of their Kronecker model. Their investigation has led to the discovery of new orthogonal matrices such as generalized Hadamard, generalized Walsh, and generalized Haar transforms.

The advantage of employing redundant factors was summarized by Andrews (1970): suppose that a matrix of order p^n by p^n can be expressed as a product of n sparse matrices each of p^2 non-redundant entries. When the original matrix is multiplied by a vector of size p^n, only np^{n+1} multiplications and the same number of additions are necessary. This compares favourably with the p^{2n} multiplications and additions normally required if the matrix is not decomposed.

1.4 Solution of Inverse Scattering Problem by Matrix Decomposition

The system of equations that arises from scattering theory (see Eq. (7.3.7)) may be put in the form $P'f = F'$. The vector f is the unknown, which contains as its elements the scattered-field source densities at

points (x_v,y_v,z_v), $v = 1,2,\ldots,n$, within the scattering object. The vector F' contains the scattered-field complex amplitudes at some points (K_{xu},K_{yu},K_{zu}), $u = 1,2,\ldots,n$, in the far-field region. The propagator matrix P' acts as a linear operator mapping the vector f into the vector F'. With a given set of data F' and a known propagator P', it is possible to determine f by solving the system of equations P' f = F'. The present problem is how to obtain f quickly and accurately.

Both P' and F' depend on the two independent variables K_x and K_y, whereas f does not. We are free to choose in the (K_x,K_y)-plane some particular points (K_{xu},K_{yu}), which will give P' a simplified form P (and modify F' accordingly to F) to allow a fast solution of P f = F. The f yielded by P f = F will be the same as the solution of P' f = F'.

P' has all its elements expressible in terms of the products of three periodic functions: $P_{uv} = \exp(-ix_vK_{xu})\exp(-iy_vK_{yu})\exp(-iz_vK_{zu})$. If we make x_vK_{xu} take the same values modulo 2π for all elements in any column of P, then we have created a redundant factor $\exp(-ix_vK_{xu})$. In a similar manner, if y_vK_{yu} has the same value modulo 2π in any column of P, we have a redundant factor $\exp(-iy_vK_{yu})$. These facts enable us to apply decomposition techniques to handle P.

In actual practice, let us assume that our optical problem requires an equi-spaced grid of source points. We can write $x_v = j_{1v}\Delta x$, $y_v = j_{2v}\Delta y$, $z_v = j_{3v}\Delta z$, where Δx, Δy, Δz are constants and j_1, j_2, j_3 are three sets of positive integers. In order to sample F regularly at points on (K_x, K_y) plane, we shall assign $K_{xu} = k_{1u}\Delta K_x + k_{3u}(2p\pi/\Delta x)$ and $K_{yu} = k_{2u}\Delta K_y + k_{4u}(2q\pi/\Delta y)$, where k_1, k_2, k_3, k_4, p, q are all integers and ΔK_x and ΔK_y are constants. By letting j_1, j_2, j_3 be increased row-wise lexicographically in steps of 1, and k_2, k_1, k_4, k_3 be increased column-wise lexicographically in steps of 1, P will become a matrix of row-wise Kronecker products of three smaller matrices R, S and T_m, with m = 1,2,..., where R, S, and T_m correspond to the three space-periodic functions in the elements of P.

At this juncture, Wouk pointed out that R, S, and T_m all are of the Vandermonde type and we already have quick methods for inverting them. Hence we can expect a solution of the system of equations P f = F results in the form $f = (S \otimes R)^{-1}F'' = (S^{-1} \otimes R^{-1})F''$, where F" is a vector function of T_m^{-1}.

In the course of developing our fast solution for P f = F, a number of fast algorithms were discovered for the inversion of special types of large matrices. We have found that among the Vandermonde matrices, there

are two special types whose inversion requires only conjugation or physical rotation of the matrix. It is convenient to make R and S of these types to save arithmetic operations.

We have further noted that the Kronecker product $(S^{-1} \otimes R^{-1})$ is still a matrix of considerable size so that matrix-vector multiplication $(S^{-1} \otimes R^{-1})F''$ becomes lengthy. Improvement is made by displacing side-wise the elements of vectors f and F" to form rectangularized arrays (Rect f) and (Rect F") respectively. Then system P f = F is finally transformed into Rect(Rect f) = R^{-1}(Rect F")$(S^{-1})^t$. The unknown complex scattered-field source densities f are obtained in the elements of the array Rect(Rect f).

This thesis will be presented in three parts.

Part one describes the properties of eleven types of matrix sequences which appear in our main algorithm, and develops some fast algorithms for computer inversion.

Part two deals with an application of the fast algorithms to the optical inverse scattering problem. We develop a special system of equations for the inverse scattering theory, and a complete method of solution.

Part three consists of an error analysis of the inversion of those sequences introduced in the thesis. An error bound is found for the method of solution developed in Part two.

PART ONE

FAST INVERSION OF MATRICES

II. SELF-INVERTING SEQUENCES OF MATRICES

2.1 Matrix Sequences

Definition The matrices $M_1, M_2, \ldots, M_r, M_{r+1}, \ldots, M_n$ are called a sequence if they are related by a recursion formula

$$M_{r+1} = f(M_r, M_{r-1}, M_{r-2}, \ldots, M_2, M_1). \qquad (2.1.1)$$

Of special interest to us is the simplest case of the above, in which a matrix in the sequence depends only on the matrix just before it, namely,

$$M_{r+1} = f(M_r). \qquad (2.1.2)$$

This type of sequence will be the main subject of discussion in this thesis.

Definition If all the member matrices $M_1, M_2, \ldots, M_n$ are orthogonal matrices, the sequence of matrices will be called an orthogonal sequence of matrices.

Definition The inverse of a sequence is defined as the sequence of inverse matrices.

Theorem 2.1 The inverse of an orthogonal sequence of symmetrical matrices is just the sequence itself.

Proof: Since the sequence consists only of orthogonal symmetrical matrices, which are self-inverting,

inverting the sequence will not change any of its member matrices.

A Generalized Definition In this thesis, a property modifier will be attached to a sequence if the modifier describes the common properties of its member matrices.

An example is the term "orthogonal sequence" defined above. This term means "a sequence of orthogonal matrices". Other terms to appear later in this thesis will be "inverted", "symmetrical", "quasi-orthogonal", "product of" sequences, etc., which will respectively contain the meaning of sequences of "inverted", "symmetrical", "quasi-orthogonal", "product of" member matrices, etc.

Matrix Notations In this thesis, it is often necessary to break up a matrix into its components, or to rotate it by 90 degrees, or to turn a matrix up side down. For the sake of convenience, the following notations will be adopted:

$M(n,m)$ represents a matrix M of dimension $n \times m$.

$M(n)$ will be used in the place of $M(n,n)$ for short.

$M(\bar{n})$ will indicate a row vector of n elements.

$M(n')$ will represent a column vector of n elements.

M^t means the transpose of M.

M^{-t} means the transpose of the inverse of M, i.e., $(M^{-1})^t$.

$\tilde{M}$ indicates the matrix M being turned upside down.

$\hat{M}$ indicates the matrix M being physically rotated clockwise through an angle of 90 degrees.

For example,

$$M(2,3) = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{pmatrix}$$

$$\tilde{M}(2,3) = \begin{pmatrix} m_{21} & m_{22} & m_{23} \\ m_{11} & m_{12} & m_{13} \end{pmatrix}$$

$$\hat{M}(3,2) = \begin{pmatrix} m_{21} & m_{11} \\ m_{22} & m_{12} \\ m_{23} & m_{13} \end{pmatrix}$$

Whenever possible, we shall use lower-case letters a,b,c,... to denote elements of a matrix. These small letters will be subscripted or double-subscripted to indicate the row and column where an element stands in a vector or in a matrix. The capital letters A,B,C,... will be used to denote a matrix or vector, and will be subscripted if they are rows or columns of a matrix. Thus, following the above examples, we have

$$M_1(\bar{3}) = [m_{11}\ m_{12}\ m_{13}], \qquad M_2(2') = \begin{pmatrix} m_{12} \\ m_{22} \end{pmatrix}.$$

The double subscripts will be written simply side by side, as in m_{22}, to save typing space. Whenever confusion may arise, a comma will be used as a separator, as in $m_{11,2}$. When a variable, e.g. r, is subscripted with another subscripted variable, e.g. k_2, J_1 or J_2, the subscript of the second variable or variables will be placed along side for convenience in typing. Thus r_{k_2} will be typed as r_{k2}, $r_{J_1J_2}$ as $r_{J1,J2}$, etc.

Furthermore, a prefix subscript attached to a matrix or vector notation would indicate the label of a numbered matrix or vector. Hence

${}_1M(2,3)$ means matrix No. 1 of size (2,3).

${}_4M_2(\bar{3})$ means the second row of matrix No. 4, with 3 elements in this row matrix.

${}_2M_3(2')$ means the third column of matrix No.2, with a size of 2 elements.

2.2 The I' Sequence

Definition Sequence I' is the sequence generated by the recursion formula

$$I'(n+1) \equiv \begin{bmatrix} O(n') & I'(n) \\ 1 & O(\bar{n}) \end{bmatrix} \qquad n = 1,2,3,.... \quad (2.2.1)$$

with $I'(1) \equiv [1]$.

Sequence I' is also known as the "Exchange Matrix" Sequence.

Properties

1. From the definition, it is obvious that Sequence I' is a symmetrical sequence. It has its

member matrices I'(n) equal to the mirror image of the identity matrices I(n). Thus,

$$I'(n) = \begin{bmatrix} 0 & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & 0 \end{bmatrix} \tag{2.2.2}$$

$$I'(n) = I'^{t}(n)$$

with t indicating a transpose.

2. I' is an orthogonal sequence

$$I'(n)I'^{t}(n) = I(n) . \tag{2.2.3}$$

3. The above indicates that the square of any member matrix of I' is an identity matrix.

$$[I'(n)]^2 = I(n) \tag{2.2.4}$$

4. From Theorem 2.1, it is clear that Sequence I' is self-inverting.

$$I'^{-1}(n) = I'(n) \tag{2.2.5}$$

5. Multiplication by Sequence I' from left will turn a sequence up side down. Let the sequence be M. Then

$$I'(n)\ M(n) = \hat{M}(n) . \tag{2.2.6}$$

For example:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 4 & 5 \\ 2 & 3 \end{bmatrix} \quad . \tag{2.2.7}$$

6. Multiplication from right by Sequence I' will turn any sequence member left to right. For example,

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 5 & 4 \end{bmatrix} \quad . \tag{2.2.8}$$

From property 4 above we have obtained our

Fast Algorithm:

The inverse of Sequence I' is just the Sequence I' itself.

2.3 The A Sequence

Definition An A sequence is one which satisfies the recursion formula

$$A(k+m) \equiv \begin{bmatrix} A(k) & 0 \\ 0 & I'(m) \end{bmatrix} \tag{2.3.1}$$

with $A(1) \equiv [1]$, and both k and m are positive integers.

Theorem 2.3.1 Sequence A is a symmetrical sequence.

$$A(n) = A^t(n) \tag{2.3.2}$$

Proof: I'(m) is symmetrical according to (2.2.2). If A(k) is symmetrical then (2.3.1) shows that A(k+m)

will also be symmetrical. Since the sequence starts with A(1) = [1] which is symmetrical, by mathematical induction, all member matrices in the sequence will be symmetrical.

Theorem 2.3.2 A(n) is an orthogonal sequence.

$$A(n)A^t(n) = I(n) \quad . \qquad (2.3.3)$$

Proof: From (2.3.2) and (2.3.1), with $n = k + m$,

$$A(n)A^t(n) = A^2(n) = \begin{bmatrix} A^2(k) & O \\ O & I'^2(m) \end{bmatrix} \qquad (2.3.4)$$

From (2.2.4), $I'^2(m) = I(m)$ which is an identity matrix. Provided A(k) is orthogonal, from (2.3.2), $A^2(k) = A(k)A^t(k) = I(k)$ which is also an identity matrix. (2.3.4) becomes

$$A(n)A^t(n) = \begin{bmatrix} I(k) & O \\ O & I(m) \end{bmatrix} = I(k+m) = I(n). \qquad (2.3.5)$$

Therefore A(n) is orthogonal. Now A(1) = [1] is orthogonal; by mathematical induction it is established that all matrices forming Sequence A will be orthogonal.

Property of Sequence A Any member matrix A(n) of the Sequence A is a permutation matrix. Postmultiplying any matrix by A(n) means to permute the respective matrix columns. The following example illustrates the manner

of permutation:

$$A(3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad (2.3.6)$$

$$\begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} A(3) = \begin{pmatrix} 2 & 4 & 3 \\ 5 & 7 & 6 \\ 8 & 0 & 9 \end{pmatrix} \qquad (2.3.7)$$

$$A(3) \begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 4 \\ 8 & 9 & 0 \\ 5 & 6 & 7 \end{pmatrix} \qquad (2.3.8)$$

(2.3.8) indicates that premultiplying by A(n) means to permute some of the rows of a matrix.

As a special case, let k = m in (2.3.1). Then the A sequence exists only for some values of n, i.e. only for

$$n = 2^r \qquad r = \text{a positive integer} .$$

The sequence is

$$A(1) = [1]$$

$$A(2) = \left(\begin{array}{c|c} 1 & \\ \hline & 1 \end{array}\right)$$

$$A(4) = \left(\begin{array}{cc|cc} 1 & & & \\ & 1 & & \\ \hline & & 1 & \\ & & & 1 \end{array}\right) \qquad (2.3.9)$$

$$A(8) = \left[\begin{array}{cccc:cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hdashline & & & & & & & 1 \\ & & & & & & 1 & \\ & & & & & 1 & & \\ & & & & 1 & & & \end{array}\right] \qquad \text{etc.}$$

The recursion formula for this special case is

$$A(2n) = \begin{bmatrix} A(n) & O \\ O & I'(n) \end{bmatrix}$$

Theorem 2.1 gives a

Fast Algorithm The inverse of sequence A is the sequence itself.

$$A^{-1}(n) = A(n) \qquad (2.3.10)$$

III. QUASI-UNITARY SEQUENCES

3.1 Quasi-unitary Sequences

Definitions Chapter II introduced two sequences I' and A which satisfy $I'I'^{t} = I$ and $AA^{t} = I$, and $I'^{2} = I$ and $A^{2} = I$. Fast algorithms were derived for these two sequences.

To generalize the algorithms, the identity matrix I above is now replaced by a non-singular diagonal matrix D. We shall call any sequence Q having the characteristic $QQ^{t} = D$ a quasi-orthogonal sequence, having the characteristic $QQ^{H} = D$, $Q^{H} = Q^{*t}$, a quasi-unitary sequence, and having the characteristic $Q^{2} = D$ a quasi-self-inverting sequence. A quasi-self-inverting sequence is a special case of a quasi-orthogonal sequence, and a quasi-orthogonal sequence is a special case of a quasi-unitary sequence.

Their fast inversion formulas can easily be derived from the definitions

Type of sequence	Characteristic	Fast inversion algorithm
quasi-self-inverting	$Q^{2} = D$	$Q^{-1} = D^{-1}Q$, alternatively $Q^{-1} = QD^{-1}$
quasi-orthogonal	$QQ^{t} = D$ $Q^{t}Q = D$	$Q^{-1} = Q^{t}D^{-1}$ $Q^{-1} = D^{-1}Q^{t}$
quasi-unitary	$QQ^{H} = D$ $Q^{H}Q = D$	$Q^{-1} = Q^{H}D^{-1}$ $Q^{-1} = D^{-1}Q^{H}$

In order to invert a diagonal matrix D, it is only necessary to reciprocate its diagonal elements. Therefore, the algorithms in the above table indicate that to invert a sequence Q, we need only to transpose Q (or conjugate and transpose Q), and modify its rows or columns as appropriate with the reciprocals of the respective elements of D.

3.2 The Diagonal Sequence

Definition Sequence D is any sequence satisfying the recursion

$$D(k+m) = \begin{bmatrix} D(k) & O \\ O & D(m) \end{bmatrix} \qquad (3.2.1)$$

with k and m being arbitrary positive integers and D(k) and D(m) being diagonal matrices.

Properties The following properties are obvious.

1. Sequence D is a diagonal sequence.
2. Sequence D is quasi-orthogonal.
3. The inverse of D is sequence D^{-1}, another diagonal sequence with

$$d_{uu}^{-1} = 1/d_{uu} \quad . \qquad (3.2.2)$$

Fast Algorithm To invert D(n), it is only necessary to reciprocate the diagonal elements.

3.3 The B Sequence

Definition Sequence B is defined by the recursion formula

$$B(2n) \equiv \begin{bmatrix} B(n) & O \\ O & I(n) \end{bmatrix} \begin{bmatrix} I(n) & I(n) \\ I(n) & -I(n) \end{bmatrix} = \begin{bmatrix} B(n) & B(n) \\ I(n) & -I(n) \end{bmatrix} \quad (3.3.1)$$

with $B(1) = [1]$.

It follows from the definition that the B sequence consists only of those B(n) where

$$n = 2^r \qquad r = \text{a positive integer.} \quad (3.3.2)$$

Therefore

$$B(1) = [1]$$

$$B(2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$B(4) = \begin{bmatrix} B(2) & O \\ O & I(2) \end{bmatrix} \begin{bmatrix} I(2) & I(2) \\ I(2) & -I(2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad (3.3.3)$$

$$B(8) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

.... etc.

Sequence B is clearly not orthogonal.

Theorem 3.3 Sequence B is quasi-orthogonal.

Proof: From (3.3.1)

$$B(2n) = \begin{pmatrix} B(n) & 0 \\ 0 & I(n) \end{pmatrix} \begin{pmatrix} I(n) & I(n) \\ I(n) & -I(n) \end{pmatrix} = \begin{pmatrix} B(n) & B(n) \\ I(n) & -I(n) \end{pmatrix} \qquad (3.3.4)$$

Let us form a product of $B(2k)$ and $B^t(2k)$.

$$B(2k)B^t(2k) = \begin{pmatrix} B(k) & B(k) \\ I(k) & -I(k) \end{pmatrix} \begin{pmatrix} B^t(k) & I(k) \\ B^t(k) & -I(k) \end{pmatrix}$$

$$= \begin{pmatrix} 2B(k)B^t(k) & 0 \\ 0 & 2I^2(k) \end{pmatrix} \qquad (3.3.5)$$

$$= 2\begin{pmatrix} B(k)B^t(k) & 0 \\ 0 & I(k) \end{pmatrix} .$$

(3.3.5) indicates that if $B(k)B^t(k)$ is diagonal then $B(2k)B^t(2k)$ will also be diagonal. Since sequence B

starts with a diagonal matrix B(1) = [1]; it can be seen from (3.3.5) that $B(2)B^t(2), B(4)B^t(4), \ldots$ are all diagonal. Therefore sequence B is quasi-orthogonal.

Now let

$$B(n)B^t(n) = D'(n) . \tag{3.3.6}$$

Setting $k = 1,2,4,8,\ldots$ in (3.3.5), the actual value of sequence D' is calculated as follows:

$$D'(1) = [1]$$

$$D'(2) = \begin{bmatrix} 2 & \\ & 2 \end{bmatrix}$$

$$D'(4) = \begin{bmatrix} 4 & & & \\ & 4 & & \\ & & 2 & \\ & & & 2 \end{bmatrix}$$

$$D'(8) = \begin{bmatrix} 8 & & & & & & & \\ & 8 & & & & & & \\ & & 4 & & & & & \\ & & & 4 & & & & \\ & & & & 2 & & & \\ & & & & & 2 & & \\ & & & & & & 2 & \\ & & & & & & & 2 \end{bmatrix} \tag{3.3.7}$$

The recursion formula for sequence D' is

$$D'(2n) = 2\begin{bmatrix} D'(n) & O \\ O & I(n) \end{bmatrix}. \tag{3.3.8}$$

And its inverse is found in (3.2.2).

Fast Algorithm The inverse of D' can be obtained by reciprocating the diagonal elements of its member matrix. Or alternatively, reciprocating the first half of its member matrix (diagonal elements only), and replacing the diagonal elements in the second half with a constant 0.5.

$$D'^{-1}(2n) = \begin{bmatrix} \frac{1}{2}D'^{-1}(n) & O \\ O & \frac{1}{2}I(n) \end{bmatrix} \qquad (3.3.9)$$

Then the inverse of Sequence B is obtained from (3.3.6).

$$B^{-1}(n) = B^{t}(n)\ D'^{-1}(n) \qquad (3.3.10)$$

In other words, the inverse of Sequence B is the transposed sequence B with each row of the member matrices modified by the respective reciprocal of the diagonal elements of the member matrices in sequence D'.

Sequence B can also be called a non-normalized "Haar matrix" sequence, with proper permutation.

IV. VANDERMONDE SEQUENCES

4.1 Vandermonde Sequence

Definition Given a sequence of distinct numbers, real or complex, denoted by $x_1, x_2, x_3, \ldots, x_N$, a sequence of matrices, called the V sequence, can be generated by the formula

$$V(n+1) \equiv \begin{bmatrix} V(n) & X_2(n') \\ X_1(\bar{n}) & x_{n+1}^n \end{bmatrix} \qquad n = 1,2,3,\ldots,N-1 \qquad (4.1.1)$$

with

$$V(1) \equiv [1]$$

$$X_1(\bar{n}) = [x_1^n \; x_2^n \; x_3^n \; \ldots\ldots \; x_n^n]$$

$$X_2(n') = [1 \; x_{n+1} \; x_{n+1}^2 \; x_{n+1}^3 \; \ldots\ldots \; x_{n+1}^{n-1}]^t .$$

The member matrices in the V sequence are called Vandermonde matrices of non-confluent type. They take the form

$$V(n) = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ x_1 & x_2 & x_3 & \ldots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \ldots & x_n^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & \ldots & x_n^{n-1} \end{bmatrix} \qquad (4.1.2)$$

Macon and Spitzbard (1958) first derived a set of formulas relating a Vandermonde matrix with its inverse. These formulas are complicated. Parker (1964) worked out another method suitable for computer implementation. On the same basis, Traub (1966) designed two algorithms. He recommended his Algorithm I for $n \geq 4$, which requires $n(7n-9)/2$ multiplications and $5n(n-1)/2$ additions to invert a Vandermonde matrix. Ballester and Pereyra (1967) found that to solve a non-confluent Vandermonde system of equations, only $n(3n-1)/2$ multiplications and the same number of additions are required. However, using their algorithm to invert a Vandermonde matrix requires more arithmetic operations than using Traub's algorithm I. More contributions on the inversion of generalized and confluent Vandermonde matrix are due to Kan (1971), Björck and Elfving (1972), Schappelle (1972) and Goknar (1973).

In this thesis we shall follow Parker's formula and Traub's algorithm I as we believe that these are the simplest for the inversion of non-confluent Vandermonde matrices. We also develop algorithms for special Vandermonde matrices.

Inversion Method A simple method to invert a non-confluent Vandermonde matrix was due to Parker (1964). Briefly, his method is as follows:

Let a polynomial $G(x)$ be formed with roots x_v ($v = 1,2,3,\ldots,n$) equal to the elements in the second row of $V(n)$ in (4.1.2):

$$G(x) \equiv \prod_{v=1}^{n} (x_v - x) . \qquad (4.1.3)$$

Then let $g_u(x)$ be another polynomial of x such that

$$g_u(x) \equiv G(x)/(x_u - x) . \qquad (4.1.4)$$

Compute the coefficients of $-g_u(x)/G'(x_u)$, $G'(x)$ being the derivative of $G(x)$. These coefficients are the elements in row u of a matrix $M(n)$. Proceeding with $u = 1,2,3,\ldots,n$, a full matrix $M(n)$ will be formed. Then $M(n)$ is the required inverse of $V(n)$.

Proof of the method:

1. From the computation of $M(n)$ and (4.1.2), it is clear that the product $M(n)V(n)$ will be a matrix with its element in row u and column v equal to $-g_u(x_v)/G'(x_u)$.

2. From the definition of a derivative, we have

$$G'(x) = \lim_{x \to x_u} G(x)/(x-x_u) . \qquad (4.1.5)$$

The right hand side of (4.1.5) is $-g_u(x)$ according to (4.1.4). Therefore $G'(x)$ approaches $-g_u(x)$ when x approaches x_u, or

$$-g_u(x_v)/G'(x_v) = 1 \qquad \text{for } v = u . \qquad (4.1.6)$$

3. (4.1.4) indicates that

$$g_u(x_v) = 0 \quad \text{or} \quad g_u(x_v)/G'(x_v) = 0 \quad \text{for} \quad v \neq u. \tag{4.1.7}$$

4. From 1. and (4.1.6) and (4.1.7) it follows that

$$M(n)\, V(n) = I(n) . \tag{4.1.8}$$

That is to say

$$M(n) = V^{-1}(n) . \tag{4.1.9}$$

This gives a general formula for the inversion of (4.1.2):

$$V^{-1}(n) = \underline{D}(n)\, \underline{V}(n) \tag{4.1.10}$$

where $\underline{D}(n)$ is a diagonal matrix equal to

$$\underline{D}(n) = \begin{pmatrix} -\prod_{i\neq 1} (x_i - x_1)^{-1} & & & \\ & -\prod_{i\neq 2} (x_i - x_2)^{-1} & & \\ & & \cdots & \\ & & & -\prod_{i\neq n} (x_i - x_n)^{-1} \end{pmatrix} \tag{4.1.11}$$

$\underline{V}(n)$ is a square matrix whose elements are elementary symmetric functions

$$\underline{V}(n)=\begin{pmatrix} 1 & \sum_{i\neq 1}(x_i) & \sum_{i,j\neq 1}(x_ix_j)/2! \ldots & \sum_{i,j..k\neq 1}(x_ix_j..x_k)/(n-1)! \\ 1 & \sum_{i\neq 2}(x_i) & \sum_{i,j\neq 2}(x_ix_j)/2! \ldots & \sum_{i,j..k\neq 2}(x_ix_j..x_k)/(n-1)! \\ & \cdots\cdots\cdots & & \\ 1 & \sum_{i\neq n}(x_i) & \sum_{i,j\neq n}(x_ix_j)/2! \ldots & \sum_{i,j..k\neq n}(x_ix_j..x_k)/(n-1)! \end{pmatrix} \qquad (4.1.12)$$

($n-1$ factors)

As an example, take n = 4. Then

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 \\ x_1^3 & x_2^3 & x_3^3 & x_4^3 \end{pmatrix}^{-1} = \begin{pmatrix} -(x_2-x_1)^{-1}(x_3-x_1)^{-1}(x_4-x_1)^{-1} & & & \\ & -(x_1-x_2)^{-1}(x_3x_2)^{-1}(x_4-x_2)^{-1} & & \\ & & -(x_1-x_3)^{-1}(x_2-x_3)^{-1}(x_4-x_3)^{-1} & \\ & & & -(x_1-x_4)^{-1}(x_2-x_4)^{-1}(x_3-x_4)^{-1} \end{pmatrix} \times$$

$$\begin{pmatrix} 1 & x_2+x_3+x_4 & x_2x_3+x_3x_4+x_4x_2 & x_2x_3x_4 \\ 1 & x_1+x_3+x_4 & x_1x_3+x_3x_4+x_4x_1 & x_1x_3x_4 \\ 1 & x_1+x_2+x_4 & x_1x_2+x_2x_4+x_4x_1 & x_1x_2x_4 \\ 1 & x_1+x_2+x_3 & x_1x_2+x_2x_3+x_3x_1 & x_1x_2x_3 \end{pmatrix} . \qquad (4.1.13)$$

From (4.1.3), (4.1.4) and (4.1.10) we can see that

$$G'(x_i) = -\underline{d}_{ii}^{-1}$$
$$g_i(x) = \sum_j \underline{v}_{ij}\, x^j \qquad (4.1.14)$$

Fast Algorithm Traub (1966) denoted the elements of an inverted Vandermonde matrix by

$$v_{ij}^{-1} = \underline{v}_{ij}\, \underline{d}_{ii} \quad \text{(compare with 4.1.10).} \qquad (4.1.15)$$

He then suggested that any jth elementary symmetric function can be calculated by recursion formulas

$$\begin{cases} s_{m,j} = s_{m-1,j} + x_m s_{m-1,j-1} & \text{for } m = 2,3,\ldots,n \\ & \quad j = 2,3,\ldots,m,m+1. \\ s_{m,j} = 0 & \text{for } j > m+1. \\ s_{m,1} = 1 \\ s_{1,2} = x_1 \\ a_j = (-1)^j s_{n,j} \end{cases} \tag{4.1.16}$$

The calculation of all the a_j requires $n(n-1)/2$ multiplications and the same number of additions.

For each $i = 1,2,\ldots,n$, $\underline{v}_{ij}$ may be computed by

$$\begin{aligned} \underline{v}_{ij} &= x_i \underline{v}_{i,j-1} + a_j \qquad j = 2,3,\ldots,n. \\ \underline{v}_{i,1} &= 1 \end{aligned} \tag{4.1.17}$$

All the $\underline{v}_{ij}$ may be obtained with $n(n-2)$ multiplications and $n(n-1)$ additions. Then $\underline{d}_{ii}$ may be computed as below for $n(n-1)$ multiplications and same number of additions:

$$\begin{aligned} \underline{v}'_{ij} &= x_i \underline{v}'_{i,j-1} + \underline{v}_{i,j-1} \qquad j = 2,3,\ldots,n. \\ \underline{v}'_{i,1} &= 1 \qquad\qquad i = 1,2,\ldots,n. \\ \underline{d}_{ii} &= (\underline{v}'_{i,n})^{-1} \end{aligned} \tag{4.1.18}$$

Let us assume that division time is comparable to a multiplication time for the above calculations. Then

the last step in Traub's algorithm is to perform the multiplications in (4.1.15), and this needs n(n-1) multiplications (the first column of $\underline{V}(n)$ is always a unit vector). Altogether, we need n(7n-9)/2 multiplications and 5n(n-1)/2 additions. This is Traub's first algorithm for inverting a non-confluent Vandermonde matrix.

4.2 The C Sequence

Definition The C Sequence is generated by the formula

$$C(n+1) \equiv \begin{pmatrix} C(n) & X_2(n') \\ X_1(\bar{n}) & x_{n+1}^{n+1} \end{pmatrix} \qquad (4.2.1)$$

with $C(1) \equiv [x_1]$

$$X_1(\bar{n}) = [x_n \; x_n^2 \; x_n^3 \; \ldots \; x_n^n]$$

$$X_2(n') = [x_1^n \; x_2^n \; x_3^n \; \ldots \; x_n^n]^t \; .$$

A member matrix in the sequence will take the form

$$C(n) = \begin{pmatrix} x_1 & x_1^2 & x_1^3 & x_1^4 & \ldots & x_1^n \\ x_2 & x_2^2 & x_2^3 & \ldots\ldots & & x_2^n \\ x_3 & x_3^2 & \ldots\ldots\ldots & & & x_3^n \\ & \ldots\ldots\ldots\ldots & & & & \\ x_n & x_n^2 & \ldots\ldots\ldots & & & x_n^n \end{pmatrix} . \qquad (4.2.2)$$

Properties

1. Every row in C(n) is a power sequence.

2. C(n) can be factorized into two matrices, the first being a diagonal one, the second being a transposed Vandermonde.

$$C(n)=\begin{pmatrix} x_1 & & & \\ & x_2 & & \\ & & \ddots & \\ & & & x_n \end{pmatrix}\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ & & \cdots\cdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} = D(n)V^t(n) \tag{4.2.3}$$

3. The inverse of C(n) is obtained from (4.2.3), namely

$$C^{-1}(n) = [D(n)V^t(n)]^{-1} = [V^t(n)]^{-1}D^{-1}(n) = [V^{-1}(n)]^t D^{-1}(n). \tag{4.2.4}$$

Fast Algorithm The inverse of Sequence C is simply a transposed inverted Vandermonde sequence, modified by a reciprocal of the first column of its member matrix, the Vandermonde being formed by dividing each row of C(n) with the first element in each row.

An example is given below.

When n = 4, we have

$$\begin{pmatrix} x_1 & x_1^2 & x_1^3 & x_1^4 \\ x_2 & x_2^2 & x_2^3 & x_2^4 \\ x_3 & x_3^2 & x_3^3 & x_3^4 \\ x_4 & x_4^2 & x_4^3 & x_4^4 \end{pmatrix}^{-1} = [V^{-1}(4)]^t[D^{-1}(4)] = [\underline{V}^t(4)][\underline{D}(4)D^{-1}(4)] =$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ x_2+x_3+x_4 & x_1+x_3+x_4 & x_1+x_2+x_4 & x_1+x_2+x_3 \\ x_2x_3+x_3x_4+x_4x_2 & x_1x_3+x_3x_4+x_4x_1 & x_1x_2+x_2x_4+x_4x_1 & x_1x_2+x_2x_3+x_3x_1 \\ x_2x_3x_4 & x_1x_3x_4 & x_1x_2x_4 & x_1x_2x_3 \end{pmatrix}$$

$$\times \begin{pmatrix} -1/\{x_1(x_2-x_1)(x_3-x_1)(x_4-x_1)\} & & & \\ & -1/\{x_2(x_1-x_2)(x_3-x_2)(x_4-x_2)\} & & \\ & & -1/\{x_3(x_1-x_3)(x_2-x_3)(x_4-x_3)\} & \\ & & & -1/\{x_4(x_1-x_4)(x_2-x_4)(x_3-x_4)\} \end{pmatrix}$$

(4.2.5)

Pereyra (1967) warned that attention should be paid to the possible instability of Vandermonde matrices as, generally speaking, these matrices can easily become ill-conditioned. In the next two sections, two special types of matrices will be introduced which are not ill-conditioned and for which there is a faster inversion method. These are the types of Vandermonde matrices that are to be used in our optical equations.

4.3 The E Sequence

Definition Sequence E is a special case of Sequence C with

$$x_u = \exp(-\pi i(2u-1)/n) \qquad i^2 = -1, \qquad u = 1,2,\ldots n. \tag{4.3.1}$$

For $n = 2^r$ and $r = 1,2,3,4,\ldots$, the E sequence takes the following form:

$$E(1) = [x] \qquad x = \exp(-i\pi)$$

$$E(2) = \begin{bmatrix} x & -1 \\ x^3 & -1 \end{bmatrix} \qquad x = \exp(-i\pi/2)$$

$$E(4) = \begin{bmatrix} x & -i & x^3 & -1 \\ x^3 & i & x & -1 \\ x^5 & -i & x^7 & -1 \\ x^7 & i & x^5 & -1 \end{bmatrix} \qquad x = \exp(-i\pi/4)$$

$$E(8) = \begin{bmatrix} x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 & -1 \\ x^3 & x^6 & x & \cdots & & x^5 & -1 & \\ x^5 & x^2 & x^7 & \cdots & & x^3 & -1 & \\ & & \cdots\cdots\cdots & & & & & \\ x^7 & x^6 & x^5 & x^4 \cdots\cdots & & x & -1 & \end{bmatrix} \qquad x = \exp(-i\pi/8)$$

$\cdots\cdots\cdots$ (4.3.2)

Since Sequence E is a special case of Sequence C, the fast algorithm described in (4.2.4) will be applicable. However, a still faster method can be developed with the help of Sequence I':

Theorem 4.3 The product of Sequence E and its transpose is equal to the Sequence I', multiplied by n.

$$E(n)\, E^{t}(n) = n\, I'(n) \tag{4.3.3}$$

Proof: According to (4.3.1) and (4.2.2), a member matrix of the Sequence E can be written as:

$$E(n) = \begin{bmatrix} \exp(k_1) & \exp(2k_1) & \cdots & \exp(nk_1) \\ \exp(k_2) & \exp(2k_2) & \cdots & \exp(nk_2) \\ & \cdots\cdots & & \\ \exp(k_n) & \exp(2k_n) & \cdots & \exp(nk_n) \end{bmatrix}, \tag{4.3.4}$$

where

$$k_u = -i\pi(2u-1)/n.$$

The element in row u and column v of E(n) is

$$e_{uv} = \exp(vk_u) . \tag{4.3.5}$$

Let e'_{uv} be an element in the uth row and vth column of the matrix product $E(n)E^{t}(n)$. Then

$$\begin{aligned} e'_{uv} &= \sum_{r=1}^{n} \exp(rk_u)\exp(rk_v) \\ &= \sum_{r=1}^{n} \exp[r(k_u+k_v)] \\ &= \sum_{r=1}^{n} \exp[-2\pi ri(u+v-1)/n] \\ &= \begin{cases} n & \text{if } (u+v-1) = n \\ 0 & \text{if } (u+v-1) \neq n . \end{cases} \end{aligned} \tag{4.3.6}$$

The matrix product can be reconstructed from e'_{uv}:

$$E(n)\ E^t(n) = \begin{pmatrix} & & & n \\ & & n & \\ & \iddots & & \\ n & & & \end{pmatrix} = n\ I'(n). \qquad (4.3.7)$$

Therefore the theorem is established.

Section 2.2 explained that I' sequence is self-inverting. From (4.3.7) it is clear that

$$E^{-1}(n) = E^t(n)\ I'(n)/n\ . \qquad (4.3.8)$$

$E^t(n)\ I'(n)$ means to rotate the matrix E(n) clockwise through a right angle, and no multiplication is involved. This leads to our

Fast Algorithm The inverted sequence E is the original sequence rotated clockwise through a right angle and then divided by n.

$$E^{-1}(n) = \hat{E}(n)/n \qquad (4.3.9)$$

the notation $\hat{E}$ being defined in Section 2.1.

A numerical example below illustrates the algorithm.

Example Given $e = \exp(-i\pi/4)$ and

$$E(4) = \begin{pmatrix} e & -i & e^3 & -1 \\ e^3 & i & e & -1 \\ -e & -i & -e^3 & -1 \\ -e^3 & i & -e & -1 \end{pmatrix} \qquad (4.3.10)$$

then

$$E^{-1}(4) = \frac{1}{4}\begin{pmatrix} -e^3 & -e & e^3 & e \\ i & -i & i & -i \\ -e & -e^3 & e & e^3 \\ -1 & -1 & -1 & -1 \end{pmatrix} \qquad (4.3.11)$$

Property of Sequence E The transposed inverted E sequence is the sequence E being turned upsidedown and divided by n.

$$(E^{-1}(n))^t = \tilde{E}(n)/n \ . \qquad (4.3.12)$$

Proof: Since from (2.2.6) $I'(n)E(n) = \tilde{E}(n)$. Therefore, from (4.3.9),

$$(E^{-1}(n))^t = (E^t(n)I'(n)/n)^t = I'(n)E(n)/n = \tilde{E}(n)/n \ . \qquad (4.3.13)$$

4.4 The Fourier Transform Sequence

Let

$$w \equiv \exp(-2\pi i/n)$$

$$x_u = w^u, \quad u = 1,2,3,....n.$$

Then (4.2.2) becomes

$$\underline{F}(n) \equiv \begin{pmatrix} w & w^2 & w^3 & \dots & w^n \\ w^2 & w^4 & w^6 & \dots & w^{2n} \\ w^3 & w^6 & w^9 & \dots & w^{3n} \\ \dots & \dots & \dots & \dots & \dots \\ w^n & w^{2n} & & \dots & w^{nn} \end{pmatrix} \qquad (4.4.1)$$

The $\underline{F}(n)$ here is usually referred to as a Fourier transform matrix. The sequence $\underline{F}$ can be formally defined as follows:

Definition The Fourier transform sequence $\underline{F}$ is a sequence generated by

$$\underline{F}(n+1) \equiv \begin{pmatrix} F(n) & X_2(n') \\ X_1(\bar{n}) & w^{(n+1)(n+1)} \end{pmatrix} \qquad (4.4.2)$$

with

$$\underline{F}(1) \equiv [w]$$

$$X_1(\bar{n}) = [w^{n+1}\ w^{2n+2} \ldots w^{n(n+1)}]$$

$$X_2(n') = [w^{n+1}\ w^{2n+2} \ldots w^{n(n+1)}]^t .$$

As far as inversion is concerned we can expect $\underline{F}^{-1}(n)$ to be derived from (4.2.4). But a still faster formula follows the definition of an inverse Fourier transform. Now we find the inverse of $\underline{F}(n)$ to be

$$\underline{F}^{-1}(n) = \frac{1}{n}\begin{pmatrix} w^{-1} & w^{-2} & \ldots & w^{-n} \\ w^{-2} & w^{-4} & \ldots & w^{-2n} \\ w^{-3} & w^{-6} & \ldots & w^{-3n} \\ & \ldots\ldots\ldots & & \\ w^{-n} & w^{-2n} & \ldots & w^{-nn} \end{pmatrix} = \underline{F}^*(n)/n \ . \qquad (4.4.3)$$

Proof: Since the (u,v)th element of $\underline{F}(n)$ is w^{uv} (see 4.4.1) and the (u,v)th element of $\underline{F}^*(n)$ is w^{-uv} (see 4.4.3) the (u,v)th element in $\underline{F}(n)\underline{F}^*(n)$ will be

$$\sum_{k=1}^{n} w^{uk}\, w^{-kv} \qquad (4.4.4)$$

$$= \sum_{k=1}^{n} w^{k(u-v)} = \begin{cases} n & \text{for all } u=v \\ 0 & \text{for all } u\neq v, \text{ since w is cyclic.} \end{cases}$$

Therefore $\underline{F}(n)\underline{F}^{*}(n) = I(n)n$, or $\underline{F}^{-1}(n) = F^{*}(n)/n$.

Fast Algorithm To invert Sequence $\underline{F}$, it is only necessary to conjugate the whole sequence $\underline{F}$ and to divide the result by n, n being the size of member matrix $\underline{F}(n)$. The inversion formula is $\underline{F}^{-1}(n) = F^{*}(n)/n$.

Property of Sequence $\underline{F}$ From (4.4.1) and (4.4.3), it is clear that both $\underline{F}$ and $\underline{F}^{-1}$ sequences are symmetrical, hence

$$\underline{F} = \underline{F}^{t} \qquad (4.4.5)$$

$$\underline{F}^{-1} = \underline{F}^{-t} \quad .$$

V. COMPOSITE SEQUENCES

5.1 The H Sequence

Definition Sequence H is a sequence formed from sequence E as follows:

$$H(2n) \equiv \begin{bmatrix} H(n) & O \\ O & E(n) \end{bmatrix} \begin{bmatrix} I(n) & I(n) \\ -I(n) & I(n) \end{bmatrix} = \begin{bmatrix} H(n) & H(n) \\ -E(n) & E(n) \end{bmatrix} \quad (5.1.1)$$

with $H(1) = [1]$ and $E(n)$ defined in (4.3.2).

Properties

1. The H Sequence consists only of matrices of the size $n = 2^r$, r being a positive integer. The first four matrices in Sequence H are listed below:

$$H(1) = [1]$$

$$H(2) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$H(4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ i & 1 & -i & -1 \\ -i & 1 & i & -1 \end{bmatrix} \quad (5.1.2)$$

$$H(8) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ i & 1 & -i & -1 & i & 1 & -i & -1 \\ -i & 1 & i & -1 & -i & 1 & i & -1 \\ -x & i & -x^3 & 1 & x^5 & -i & x^7 & -1 \\ -x^3 & -i & -x & 1 & x^7 & i & x^5 & -1 \\ -x^5 & i & -x^7 & 1 & x & -i & x^3 & -1 \\ -x^7 & -i & -x^5 & 1 & x^3 & i & x & -1 \end{bmatrix}, \quad x = \exp(-i\pi/4).$$

2. If we form the product of $H(n)$ and $H^t(n)$ using the formula in (5.1.1), then

$$H(n)H^t(n) = \begin{bmatrix} H(n/2) & H(n/2) \\ -E(n/2) & E(n/2) \end{bmatrix} \begin{bmatrix} H^t(n/2) & -E^t(n/2) \\ H^t(n/2) & E^t(n/2) \end{bmatrix} \tag{5.1.3}$$

$$= \begin{bmatrix} 2H(n/2)H^t(n/2) & 0 \\ 0 & 2E(n/2)E^t(n/2) \end{bmatrix},$$

and from (4.3.7)

$$H(n)H^t(n) = \begin{bmatrix} 2H(n/2)H^t(n/2) & 0 \\ 0 & nI'(n/2) \end{bmatrix}. \tag{5.1.4}$$

Setting $n = 1,2,4,\ldots$, we obtain

$$H(1)H^t(1) = [1]$$

$$H(2)H^t(2) = \left[\begin{array}{c|c} 2 & \\ \hline & 2 \end{array}\right]$$

$$H(4)H^t(4) = \begin{pmatrix} 4 & & & \\ & 4 & & \\ & & 4 & \\ & & & 4 \end{pmatrix} \qquad (5.1.5)$$

$$H(8)H^t(8) = \begin{pmatrix} 8 & & & & & & & \\ & 8 & & & & & & \\ & & 8 & & & & & \\ & & & 8 & & & & \\ & & & & 8 & & & \\ & & & & & 8 & & \\ & & & & & & 8 & \\ & & & & & & & 8 \end{pmatrix}$$

Theorem 5.2 The product of Sequence H and its transpose is equal to the Sequence A multiplied by n:

$$H(n)H^t(n) = nA(n) \ . \qquad (5.1.6)$$

Proof: When n = 1, $H(1)H^t(1) = [1]$ and $1A(1) = [1]$. Hence (5.1.6) holds for n = 1.

If (5.1.6) is true for n = k, according to (5.1.4) and (2.3.1)

$$H(2k)H^t(2k) = \begin{pmatrix} 2H(k)H^t(k) & O \\ O & 2kI'(k) \end{pmatrix} = \begin{pmatrix} 2kA(k) & O \\ O & 2kI'(k) \end{pmatrix}$$

$$= 2k \begin{pmatrix} A(k) & O \\ O & I'(k) \end{pmatrix} = 2k\ A(2k) \ . \qquad (5.1.7)$$

That is to say, (5.1.6) is also true for n = 2k. Hence the theorem is established for all $n = 2^r$ (r being a positive integer).

Fast Algorithm From (5.1.6)

$$H^{-1}(n) = H^{t}(n)A^{-1}(n)/n.$$

But (2.3.10) shows that $A^{-1}(n) = A(n)$, therefore

$$H^{-1}(n) = H^{t}(n)A(n)/n. \qquad (5.1.8)$$

The inverse of H sequence is a transposed H sequence multiplied by Sequence A and divided by n.

Since multiplying by Sequence A means permuting some of the matrix columns of H^{t}, inverting Sequence H is identical to transposing H, dividing by n, and then permuting some of the columns.

Starting from the next section, the operator ⊗ is to be employed to simplify our statements for the remaining of the thesis. A ⊗ B means a Kronecker product performed with the two matrices A and B. In Appendix I, the properties of a Kronecker operator is described for easy reference.

5.2 Kronecker Sequence

There are some sequences which appear difficult to invert when taken as a whole, but an analytical study of the sequence may disclose that it is composed from several simpler sequences. In such a case it may be possible to invert the component sequences one by one as partial solutions, then piece together the partial

solutions in some way to obtain the inverse of the original sequence. The rest of this chapter will give examples which illustrate the method. The next sections will introduce sequences L and W. These are the sequences to appear rather frequently in Part Two of the thesis. The general form of the sequences will be studied here.

Let us consider a sequence whose member matrix takes the form (see Section 2.1 for notation of subscripts)

$$L(n)=\begin{vmatrix} r_{11}s_{11} & r_{12}s_{11}\cdots r_{1,J1}s_{11} & r_{11}s_{12} & r_{12}s_{12}\cdots r_{1,J1}s_{1,J2} \\ r_{21}s_{11} & r_{22}s_{11}\cdots r_{2,J1}s_{11} & r_{21}s_{12} & r_{22}s_{12}\cdots r_{2,J1}s_{1,J2} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ r_{J1,1}s_{11} & r_{J1,2}s_{11} & \cdots\cdots & r_{J1,J1}s_{1,J2} \\ r_{11}s_{21} & r_{12}s_{21} & \cdots\cdots & r_{1,J1}s_{2,J2} \\ r_{21}s_{21} & r_{22}s_{21} & \cdots\cdots & r_{2,J1}s_{2,J2} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ r_{J1,1}s_{J2,1} & r_{J1,2}s_{J2,2} & \cdots\cdots & r_{J1,J1}s_{J2,J2} \end{vmatrix} \qquad (5.2.1)$$

The matrix contains only those elements which are the products of r_{ij} and s_{uv}, and r and s are respectively the elements of sequences R and S where

$$R(J_1) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1,J1} \\ r_{21} & r_{22} & r_{23} & \cdots & r_{2,J1} \\ & \cdots\cdots\cdots & & & \\ r_{J1,1} & r_{J1,2} & r_{J1,3} & \cdots & r_{J1,J1} \end{bmatrix} \tag{5.2.2}$$

$$S(J_2) = \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdots & s_{1,J2} \\ s_{21} & s_{22} & s_{23} & \cdots & s_{2,J2} \\ & \cdots\cdots\cdots & & & \\ s_{J2,1} & s_{J2,2} & s_{J2,3} & \cdots & s_{J2,J2} \end{bmatrix}$$

then the sequence described in (5.2.1) can also be defined by the

Definition Sequence L is a Kronecker product of two sequences R and S.

$$L(J_1J_2) \equiv S(J_2) \otimes R(J_1) . \tag{5.2.3}$$

According to the properties given in Appendix I, we have

$$L^{-1}(J_1J_2) = S^{-1}(J_2) \otimes R^{-1}(J_1) . \tag{5.2.4}$$

Fast Algorithm The inverse of an L sequence can be obtained in two steps: first, invert the factor matrices R and S individually; second, form the Kronecker product of the inverted R and S. The Kronecker product thus obtained is the inverted L sequence.

(5.2.4) requires only $J_1^2 J_2^2$ multiplications, plus the necessary operations to invert R and S individually.

5.3 Row-wise Kronecker Sequence

Let a sequence W be constructed in the following manner:

1. It consists only of those matrices W(n) with

$n = J_1 J_2$ $\quad$ J_1, J_2 being positive integers.

2. The elements of W(n) contain only power products of two complex sets of numbers t and z, where

$$t = \{t_1, t_2, t_3, \ldots\ldots\ldots t_{J1J2}\}$$

$$z = \{z_1, z_2, z_3, \ldots\ldots\ldots z_{J1}\}$$

$$W(J_1 J_2) =$$

$$\begin{bmatrix}
z_1t_1 & z_1^2t_1 & z_1^3t_1 \ldots z_1^{J1}t_1 & z_1t_1^2 & z_1^2t_1^2 \ldots z_1^{J1}t_1^2 & z_1t_1^3 & z_1^2t_1^3 \ldots z_1^{J1}t_1^{J2} \\
z_1t_2 & z_1^2t_2 \ldots\ldots z_1^{J1}t_2 & z_1t_2^2 \ldots\ldots & \ldots\ldots & & & z_1^{J1}t_2^{J2} \\
\ldots\ldots & \ldots\ldots & & & & & \ldots\ldots \\
z_1t_{J2} & z_1^2t_{J2} & \ldots\ldots & & & & z_1^{J1}t_{J2}^{J2} \\
z_2t_{J2+1} & & \ldots\ldots & & & & z_2^{J1}t_{J2+1}^{J2} \\
\ldots\ldots & & \ldots\ldots & & & & \ldots\ldots \\
z_{J1}t_{J1J2} & & & & & & z_{J1}^{J1}t_{J1J2}^{J2}
\end{bmatrix} \qquad (5.3.1)$$

Here the subscripts and superscripts follow the notations stated in Section 2.1.

3. The t has one value in each row of $W(J_1J_2)$ but the value varies from row to row.

4. The z has the same value in each row and also the same value in consecutive sets of J_2 rows. The value of z changes after every J_2 rows.

5. The powers are so arranged that the powers of t are kept at one for the first J_1 columns, and increase by one every J_1 columns. The powers of z increment by one from column to column until the powers of z reach a maximum of J_1, then the powers of z start from one again and continue to increment by one from column to column.

A careful study of (5.3.1) discloses that the above descriptions can be simplified to the

Definition Sequence W is a sequence consisting of row-wise Kronecker products of sequence Z and sequences ${}_uT$, $u = 1,2,3,\ldots,J_1$:

$$W(J_1J_2) \equiv \begin{pmatrix} {}_1T(J_2) \otimes Z_1(\overline{J_1}) \\ {}_2T(J_2) \otimes Z_2(\overline{J_1}) \\ \cdots\cdots\cdots \\ {}_{J1}T(J_2) \otimes Z_{J1}(\overline{J_1}) \end{pmatrix}. \qquad (5.3.2)$$

Here $Z_u(\overline{J_1})$ is the uth row of matrix $Z(J_1)$:

$$Z(J_1) \equiv \begin{pmatrix} z_1(\overline{J_1}) \\ z_2(\overline{J_1}) \\ \cdots \\ z_{J1}(\overline{J_1}) \end{pmatrix} \equiv \begin{pmatrix} z_1 & z_1^2 & z_1^3 & \cdots & z_1^{J1} \\ z_2 & z_2^2 & z_2^3 & \cdots & z_2^{J1} \\ & & \cdots\cdots & & \\ z_{J1} & z_{J1}^2 & & \cdots & z_{J1}^{J1} \end{pmatrix}, \qquad (5.3.3)$$

The ${}_uT(J_2)$ are defined as:

$${}_1T(J_2) \equiv \begin{pmatrix} t_1 & t_1^2 & t_1^3 & \cdots & t_1^{J2} \\ t_2 & t_2^2 & t_2^3 & \cdots & t_2^{J2} \\ & & \cdots\cdots & & \\ t_{J2} & t_{J2}^2 & t_{J2}^3 & \cdots & t_{J2}^{J2} \end{pmatrix} \qquad (5.3.4a)$$

$${}_2T(J_2) \equiv \begin{pmatrix} t_{J2+1} & t_{J2+1}^2 & \cdots & t_{J2+1}^{J2} \\ t_{J2+2} & t_{J2+2}^2 & \cdots & t_{J2+2}^{J2} \\ & \cdots\cdots & & \\ t_{2J2} & t_{2J2}^2 & \cdots & t_{2J2}^{J2} \end{pmatrix} \qquad (5.3.4b)$$

$${}_{J1}T(J_2) \equiv \begin{pmatrix} t_{(J1-1)J2+1} & \cdots & t_{(J1-1)J2+1}^{J2} \\ t_{(J1-1)J2+2} & \cdots & t_{(J1-1)J2+2}^{J2} \\ \cdots\cdots & & \\ t_{J1J2} & t_{J1J2}^2 & \cdots & t_{J1J2}^{J2} \end{pmatrix} \qquad (5.3.4c)$$

Fast Algorithm We assert that W Sequence can be inverted in three steps:

1. Form $Z(J_1)$ as shown in (5.3.3), and invert it to obtain $Z^{-1}(J_1)$.

Let

$$Z^{-1}(J_1) = [Z_1^{-1}(J_1') \;\; Z_2^{-1}(J_1') \;\; \ldots\ldots \;\; Z_{J1}^{-1}(J_1')]. \qquad (5.3.5)$$

2. Form all ${}_uT(J_2)$ as in (5.3.4) and invert them to yield ${}_1T^{-1}(J_2)$, ${}_2T^{-1}(J_2)$, $\ldots\ldots$, ${}_{J1}T^{-1}(J_2)$.

3. A suitable arrangement of Kronecker products of the inverted components will give an inverted sequence W^{-1}:

$$W^{-1}(J_1J_2) = [{}_1T^{-1}(J_2) \otimes Z_1^{-1}(J_1') \;\; {}_2T^{-1}(J_2) \otimes Z_2^{-1}(J_1') \ldots$$
$$\ldots \; {}_{J1}T^{-1}(J_2) \otimes Z_{J1}^{-1}(J_1')]. \qquad (5.3.6)$$

Proof of the Fast Algorithm For the sake of easy explanation, let us form a linear system of equations of order J_1J_2.

$$W(J_1J_2)\; X(J_1J_2') = Y(J_1J_2') \qquad (5.3.7)$$

where W is defined by (5.3.2) and

$$X(J_1J_2') = [x_1 \;\; x_2 \;\; \ldots\ldots \;\; x_{J1J2}]^t$$
$$Y(J_1J_2') = [y_1 \;\; y_2 \;\; \ldots\ldots \;\; y_{J1J2}]^t . \qquad (5.3.8)$$

Our aim is to solve for $X = U\,Y$ and to prove that U will be identical to W^{-1} in (5.3.6).

Our proof starts by substituting (5.3.2) into (5.3.7) giving

$$\begin{bmatrix} {}_1T(J_2)\otimes Z_1(\overline{J_1}) \\ \\ {}_2T(J_2)\otimes Z_2(\overline{J_1}) \\ \\ \cdots\cdots\cdots \\ \\ \cdots\cdots\cdots \\ \\ {}_{J1}T(J_2)\otimes Z_{J1}(\overline{J_1}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ x_{J1J2} \end{bmatrix} = \begin{bmatrix} (y_1\ y_2\ \cdots\ y_{J2})^t \\ \\ (y_{J2+1}\ \cdots\ y_{2J2})^t \\ \\ \cdots\cdots\cdots \\ \\ \cdots\cdots\cdots \\ \\ (y_{(J1-1)J2+1}\cdots y_{J1J2})^t \end{bmatrix} \tag{5.3.9}$$

The right hand side of (5.3.9) has a column vector Y of J_1J_2 elements. These elements are split up into J_1 groups of J_2 elements each. The <u>first group</u> only relates to ${}_1T(J_2) \otimes Z_1(\overline{J_1})$ in (5.3.9):

$$[{}_1T(J_2)\otimes Z_1(\overline{J_1})]X(J_1J_2') = [y_1\ y_2\ \cdots\ y_{J2}]^t$$

or

$$[{}_1T(J_2)][I(J_2)\otimes Z_1(\overline{J_1})]X(J_1J_2') = [y_1\ y_2\ \cdots\ y_{J2}]^t\ .$$

Inverting ${}_1T(J_2)$ to obtain ${}_1M(J_2)$:

$${}_1T^{-1}(J_2) \equiv {}_1M(J_2) \equiv [{}_1M_1(\overline{J_2})\ {}_2M_1(\overline{J_2})\ \cdots\ {}_{J2}M_1(\overline{J_2})]^t. \tag{5.3.10}$$

Then

$$[I(J_2)\otimes Z_1(\overline{J_1})]X(J_1J_2') = [{}_1M_1(\overline{J_2})\ {}_2M_1(\overline{J_2})\cdots{}_{J2}M_1(\overline{J_2})]^t$$
$$[y_1\ y_2\ \cdots\ y_{J2}]^t\ . \tag{5.3.11}$$

$[I(J_2) \otimes Z_1(\overline{J_1})]$ is a checker-board matrix; each row (i.e. each $Z_1(\overline{J_1})$) operates on a different set of elements of X. It is possible to split (5.3.11) into J_2 equations as follows:

$$[Z_1(\overline{J_1})][x_1\ x_2 \ldots x_{J1}]^t = [{}_1M_1(\overline{J_2})][y_1 \ldots y_{J2}]^t \qquad (5.3.12a)$$

$$[Z_1(\overline{J_1})][x_{J1+1} \ldots x_{2J1}]^t = [{}_2M_1(\overline{J_2})][y_1 \ldots y_{J2}]^t \qquad (5.3.12b)$$

$$\ldots\ldots\ldots\ldots$$

$$[Z_1(\overline{J_1})][x_{(J2-1)J1+1} \ldots x_{J1J2}]^t = [{}_{J2}M_1(\overline{J_2})][y_{2J2+1} \ldots y_{2J2}]^t. \qquad (5.3.12c)$$

In the meantime, we have a second group of elements on the right hand side of (5.3.9) relating to ${}_2T(J_2) \otimes Z_2(\overline{J_1})$ in (5.3.9). By the same derivation as above we obtain another J_2 equations corresponding to (5.3.12):

$$[Z_2(\overline{J_1})][x_1\ x_2 \ldots x_{J1}]^t = [{}_1M_2(\overline{J_2})][y_{J2+1} \ldots y_{2J2}]^t \qquad (5.3.13a)$$

$$[Z_2(\overline{J_1})][x_{J1+1} \ldots x_{2J1}]^t = [{}_2M_2(\overline{J_2})][y_{J2+1} \ldots y_{2J2}]^t \qquad (5.3.13b)$$

$$\ldots\ldots\ldots\ldots$$

$$[Z_2(\overline{J_1})][x_{(J2-1)J1+1} \ldots x_{J1J2}]^t = [{}_{J2}M_2(\overline{J_2})][y_{J2+1} \ldots y_{2J2}]^t. \qquad (5.3.13c)$$

All other groups of elements will be handled in a similar way.

Then we list (5.3.12a), (5.3.13a) and the first equations of all other groups together to yield

$$\begin{pmatrix} z_1(\overline{J_1}) \\ z_2(J_1) \\ \cdots \\ z_{J1}(\overline{J_1}) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_{J1} \end{pmatrix} = \begin{pmatrix} {}_1M_1(\overline{J_2}) & & & \\ & {}_1M_2(\overline{J_2}) & & \\ & & \cdots\cdots & \\ & & & {}_1M_{J1}(\overline{J_2}) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_{J1J2} \end{pmatrix}$$

(5.3.14)

Therefore, referring to (5.3.3),

$$[x_1\ x_2 \ldots x_{J1}]^t = [z^{-1}(J_1)] \begin{pmatrix} {}_1M_1(\overline{J_2}) & & & \\ & {}_1M_2(\overline{J_2}) & & \\ & & \cdots & \\ & & & {}_1M_{J1}(\overline{J_2}) \end{pmatrix} Y(J_1J_2').$$

(5.3.14a)

Similarly, putting all the second equations of each group together will give

$$[x_{J1+1} \ldots x_{2J1}]^t = [z^{-1}(J_1)] \begin{pmatrix} {}_2M_1(\overline{J_2}) & & & \\ & {}_2M_2(\overline{J_2}) & & \\ & & \cdots & \\ & & & {}_2M_{J1}(\overline{J_2}) \end{pmatrix} Y(J_1J_2').$$

(5.3.14b)

Combining (5.3.14a), (5.3.14b), etc. and referring to (5.3.10),

$$X(J_1J_2') = [{}_1M(J_2) \otimes z_1^{-1}(J_1') \ldots {}_{J1}M(J_2) \otimes z_{J1}^{-1}(J_1')] Y(J_1J_2') .$$

(5.3.15)

The coefficient matrix above is the same as W^{-1} in (5.3.6).

Generalization T and Z were defined in (5.3.3) and (5.3.4) as Vandermonde matrices for illustration only. Readers may have noticed that in our proof of the fast algorithm, T and Z were treated as general matrices, and that the power series characteristic of Vandermonde matrices were not needed throughout our derivation. Therefore the restrictions (5.3.3) and (5.3.4) may be removed in future reference to Sequence W. That is to say: as far as W is defined as (5.3.2) and T and Z are square and regular matrices, the inverse of W will be that in (5.3.6) irrespective of the structure of T and Z.

To illustrate this point, we give below

A Numerical Example: Find the inverse of W(8), where

$$W(8) = \left[\begin{array}{cccccccc} .3 & .3 & .35 & .35 & -.3 & -.3 & -.05 & -.05 \\ -.6 & -.6 & .3 & .3 & .6 & .6 & .1 & .1 \\ -.1 & -.1 & .05 & .05 & .1 & .1 & -.15 & -.15 \\ .9 & .9 & -.45 & -.45 & .1 & .1 & .35 & .35 \\ \hline .3 & .6 & .35 & .7 & -.3 & -.6 & -.05 & .1 \\ -.6 & -1.2 & .3 & .6 & .6 & 1.2 & .1 & .2 \\ -.1 & -.2 & .05 & .1 & .1 & .2 & -.15 & -.3 \\ .9 & 1.8 & -.45 & -.9 & .1 & .2 & .35 & .7 \end{array}\right] \qquad (5.3.16)$$

Inspection shows that (1) the upper half of W(8) has its even columns identical to its odd columns; (2) in the lower half of W(8), the even columns double their preceding columns; and (3) the upper half of W(8) has its odd columns identical to the odd columns in the lower half of W(8). This leads us to rewrite W(8) as

$$W(8) = \begin{pmatrix} T(4) \otimes [1 \quad 1] \\ T(4) \otimes [1 \quad 2] \end{pmatrix} \tag{5.3.17}$$

and

$$T(4) = \begin{pmatrix} .3 & .35 & -.3 & -.05 \\ -.6 & .3 & .6 & .1 \\ -.1 & .05 & .1 & -.15 \\ .9 & -.45 & .1 & .35 \end{pmatrix} . \tag{5.3.18}$$

We now form Z(2) according to the two row vectors in (5.3.17).

$$Z(2) = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} . \tag{5.3.19}$$

Z(2) and T(4) are <u>not</u> Vandermonde matrices. Inverting Z and T by usual methods yields

$$Z^{-1}(2) = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \tag{5.3.20}$$

and

$$T^{-1}(4) = \begin{pmatrix} 1 & 0 & 2 & 1 \\ 2 & 1 & 0 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 1 & -6 & 0 \end{pmatrix} . \qquad (5.3.21)$$

From (5.3.6), $W^{-1}(8)$ can be computed using (5.3.20) and (5.3.21).

$$W^{-1}(8) = \begin{pmatrix} T^{-1}(4) \otimes \begin{pmatrix} 2 \\ -1 \end{pmatrix} & T^{-1}(4) \otimes \begin{pmatrix} -1 \\ 1 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} 2 & 0 & 4 & 2 & -1 & 0 & -2 & -1 \\ -1 & 0 & -2 & -1 & 1 & 0 & 2 & 1 \\ 4 & 2 & 0 & 0 & -2 & -1 & 0 & 0 \\ -2 & -1 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 2 & 6 & 2 & 0 & -1 & -3 & -1 \\ 0 & -1 & -3 & -1 & 0 & 1 & 3 & 1 \\ 0 & 2 & -12 & 0 & 0 & -1 & 6 & 0 \\ 0 & -1 & 6 & 0 & 0 & 1 & -6 & 0 \end{pmatrix} \qquad (5.3.22)$$

It is easy to verify $W\,W^{-1} = I$ by multiplying (5.3.16) with (5.3.22).

VI. SYSTEMS OF EQUATIONS WITH KRONECKER COEFFICIENT MATRICES

6.1 System Solution by Matrix Inversion

Consider the system of equations

$$[S(J_2) \otimes R(J_1)]X(J_1J_2') = Y(J_1J_2') . \qquad (6.1.1)$$

The technique developed in (5.2.4) shows that (6.1.1) has the solution

$$X(J_1J_2') = [S^{-1}(J_2) \otimes R^{-1}(J_1)]Y(J_1J_2') . \qquad (6.1.2)$$

Suppose R and S belong to the E sequence so that $R^{-1}(J_1)$ and $S^{-1}(J_2)$ are simply $\hat{R}(J_1)/J_1$ and $\hat{S}(J_2)/J_2$ (see 4.3.9). Then

$$X(J_1J_2') = [\hat{S}(J_2) \otimes \hat{R}(J_1)]Y(J_1J_2')/(J_1J_2) . \qquad (6.1.3)$$

Now we estimate the number of operations needed for computing X.

To perform the arithmetic of (6.1.3) requires

$J_1^2J_2^2$ multiplications to yield the $J_1J_2 \times J_1J_2$ matrix $[\hat{S} \otimes \hat{R}]$,

$(J_1J_2)^2$ more multiplications and additions to get $[\hat{S} \otimes \hat{R}]Y$,

J_1J_2 divisions to obtain X.

Additions usually need much less time than multiplications. The CPU time required for additions may be ignored for the sake of simplicity.

$$\text{Total operations} = 2(J_1J_2)^2 + J_1J_2 \approx 2(J_1J_2)^2 \qquad (6.1.4)$$

if J_1, J_2 are large.

This indicates that using matrix inversion to solve (6.1.1) has a distinct advantage over using Gaussian Elimination, which normally needs $(J_1J_2)^3/3$ operations. However, (6.1.3) does not give the quickest solution for (6.1.1). There is a still faster algorithm based on transformation of the system (see 6.2.1a). This algorithm can solve (6.1.1) in $J_1J_2(J_1+J_2)$ operations.

6.2 Solution of a 2-plex System by Rectangularizing the Unknown Vector

Definition A vector X(n') is said to be rectangularized into a matrix X(n/k,k) if its n elements are split into k groups in the original order, and the groups are displaced sidewise in k columns with n/k elements in each column*. An operator Rect_k is used for this purpose. For example,

---

*Note: We assume n divisible by k.

$$X(24') = (x_1\ x_2\ x_3\ \ldots\ x_{24})^t$$

$$\text{Rect}_4\, X = \begin{pmatrix} x_1 & x_7 & x_{13} & x_{19} \\ x_2 & x_8 & x_{14} & x_{20} \\ \multicolumn{4}{c}{\ldots\ldots\ldots} \\ x_6 & x_{12} & x_{18} & x_{24} \end{pmatrix}.$$

If X is a matrix, then to rectangularize X would mean to rectangularize each of the columns of X.*

$$\text{Rect}_2(\text{Rect}_4 X) = \begin{pmatrix} x_1 & x_4 & x_7 & x_{10} & \cdots & x_{22} \\ x_2 & x_5 & x_8 & x_{11} & \cdots & x_{23} \\ x_3 & x_6 & x_9 & x_{12} & \cdots & x_{24} \end{pmatrix}$$

In general, $\text{Rect}_k X(a,b) \equiv X(a/k,bk)$.

Definition A linear system of equations $L(n)X(n') = Y(n')$ is said to be m-plex if its coefficient matrix $L(n)$ is a Kronecker product of m matrices: $L(n) = S_1(J_1) \otimes S_2(J_2) \otimes \ldots \otimes S_m(J_m)$, $n = J_1J_2J_3\ldots J_m$.

Theorem 6.2 Provided $L(J_1J_2) = S(J_2) \otimes R(J_1)$, the 2-plex system of equations $L(J_1J_2)X(J_1J_2') = Y(J_1J_2')$ can be transformed into the following:

$$R(J_1)(\text{Rect}_{J2}\ X) = (\text{Rect}_{J2}\ Y)[S^{-1}(J_2)]^t \qquad (6.2.1)$$

where (see Section 2.1 for notation of subscripts)

---

*Note: Ekstrom (1973) defined a "stacking operator" to perform the reverse of a rectangularization process.

$$\text{Rect}_{J2}X = \begin{pmatrix} x_1 & x_{J1+1} & x_{2J1+1} & \cdots & x_{(J2-1)J1+1} \\ x_2 & x_{J1+1} & x_{2J1+2} & \cdots & x_{(J2-1)J1+2} \\ & \cdots\cdots\cdots\cdots & & & \\ x_{J1} & x_{2J1} & x_{3J1} & \cdots & x_{J1J2} \end{pmatrix} \qquad (6.2.2)$$

and $\text{Rect}_{J2}Y$ is same as $\text{Rect}_{J2}X$ except that x is replaced by y.

In other words, (6.2.1) gives the solution of the 2-plex system as (6.2.1a)

$$\text{Rect}_{J2}X = [R^{-1}(J_1)](\text{Rect}_{J2}Y)[S^{-1}(J_2)]^t. \qquad (6.2.1a)$$

Proof: Substituting (6.1.1) into L X = Y yields

$$\begin{pmatrix} s_{11}r_{11} & s_{11}r_{12} & \cdots & s_{1,J2}r_{1,J1} \\ s_{11}r_{21} & s_{11}r_{22} & \cdots & s_{1,J2}r_{2,J1} \\ \cdots\cdots & & & \\ s_{21}r_{11} & s_{21}r_{12} & \cdots & s_{2,J2}r_{1,J1} \\ s_{21}r_{21} & s_{21}r_{22} & \cdots & s_{2,J2}r_{2,J1} \\ \cdots\cdots & & & \\ s_{J2,1}r_{J1,1} & \cdots & & s_{J2,J2}r_{J1,J1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_{J1+1} \\ x_{J1+2} \\ \cdots \\ x_{J1J2} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_{J1+1} \\ y_{J1+2} \\ \cdots \\ y_{J1J2} \end{pmatrix} \qquad (6.2.3)$$

Putting the 1st, (J_1+1)th, $(2J_1+1)$th, $(3J_1+1)$th equations together and combining r with x to form a term, (6.2.3) becomes

$$\begin{pmatrix} s_{11}\ s_{11}\ s_{11}\cdots s_{12}\ s_{12}\cdots s_{1,J2} \\ s_{21}\ s_{21}\ s_{21}\cdots s_{22}\ s_{22}\cdots s_{2,J2} \\ \cdots\cdots\cdots\cdots \\ s_{J2,1}\ s_{J2,1}\cdots s_{J2,2}\ \cdots s_{J2J2} \end{pmatrix} \begin{pmatrix} r_{11}x_1 \\ r_{12}x_2 \\ \cdots\cdots \\ r_{1,J1}x_{J1J2} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_{J1+1} \\ \cdots\cdots \\ y_{(J2-1)J1+1} \end{pmatrix} \tag{6.2.4}$$

(6.2.4) has a coefficient matrix of size $J_2 \times J_1 J_2$. Since every s repeats J_1 times in the matrix, (6.2.4) can be written as

$$\begin{pmatrix} s_{11}\ s_{12}\ s_{13}\cdots s_{1,J2} \\ s_{21}\ s_{22}\ s_{23}\cdots s_{2,J2} \\ \cdots\cdots\cdots\cdots \\ s_{J2,1}\ s_{J2,2}\cdots s_{J2,J2} \end{pmatrix} \begin{pmatrix} R_1(\overline{J_1}) & & & \\ & R_1(\overline{J_1}) & & \\ & & \cdots\cdots & \\ & & & R_1(\overline{J_1}) \end{pmatrix} \begin{pmatrix} \bar{x}_1 \\ x_2 \\ \cdots \\ x_{J1J2} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_{J1+1} \\ \cdots \\ \cdots \end{pmatrix} \tag{6.2.5}$$

The matrix $S(J_2)$ in (6.2.5) can be inverted:

$$S^{-1}(J_2) = \begin{pmatrix} S_1^{-1}(\overline{J_2}) \\ S_2^{-1}(\overline{J_2}) \\ \cdots\cdots \\ S_{J2}^{-1}(\overline{J_2}) \end{pmatrix} \tag{6.2.5a}$$

Pre-multiplying (6.2.5) with (6.2.5a), the first equation of (6.2.5) is

$$R_1(\overline{J_1})(x_1\ x_2\ \cdots\ x_{J1})^t = S_1^{-1}(\overline{J_2})(y_1\ y_{J1+1}\ \cdots\ y_{(J2-1)J1+1})^t. \tag{6.2.6a}$$

Similarly, the second, third, etc. equations of (6.2.5) are

$$R_1(\overline{J_1})(x_{J1+1}\cdots x_{2J1})^t = S_2^{-1}(\overline{J_2})(y_1\ y_{J1+1}\cdots y_{(J2-1)J1+1})^t, \quad (6.2.6b)$$

$$R_1(\overline{J_1})(x_{2J1+1}\cdots x_{3J1})^t = S_3^{-1}(\overline{J_2})(y_1\ y_{J1+1}\cdots y_{(J2-1)J1+1})^t \quad (6.2.6c)$$

.....

$$R_1(\overline{J_1})(x_{(J2-1)J1+1}\cdots x_{J1J2})^t = S_{J2}^{-1}(\overline{J_2})(y_1\ y_{J1+1}\cdots \cdots y_{(J2-1)J1+1})^t. \quad (6.2.6d)$$

Then, we put the 2nd, (J_1+2)th, $(2J_1+2)$th, ... equations of (6.2.3) together, and in the same manner we shall obtain

$$R_2(\overline{J_1})(x_1\ x_2\cdots x_{J1})^t = S_1^{-1}(\overline{J_2})(y_2\ y_{J1+2}\cdots y_{(J2-1)J1+2})^t, \quad (6.2.7a)$$

$$R_2(\overline{J_1})(x_{J1+1}\cdots x_{2J1})^t = S_2^{-1}(\overline{J_2})(y_2\ y_{J1+2}\cdots y_{(J2-1)J1+2})^t. \quad (6.2.7b)$$

We can also put the 3rd, (J_1+3)th, etc. equations of (6.2.3) together and obtain another set of equations.

The next step will be to put (6.2.6a), (6.2.7a) and the first equations of all other sets of equations together. This will yield

$$R(J_1)(x_1 \ldots x_{J1})^t = \begin{pmatrix} S_1^{-1}(\overline{J_2})(y_1 \ldots y_{(J2-1)J1+1})^t \\ S_1^{-1}(\overline{J_2})(y_2 \ldots y_{(J2-1)J1+2})^t \\ \ldots\ldots\ldots\ldots\ldots \end{pmatrix}$$

$$= \text{Rect}_{J2}{}^{Y}[S_1^{-1}(\overline{J_2})]^t \quad . \qquad (6.2.8)$$

Premultiplying (6.2.8) with $R^{-1}(J_1)$,

$$(x_1 \ldots x_{J1})^t = R^{-1}(J_1)\ \text{Rect}_{J2}{}^{Y}[S_1^{-1}(\overline{J_2})]^t. \qquad (6.2.9)$$

Similarly, putting (6.2.6b), (6.2.7b) etc. together will give

$$(x_{J1+1} \ldots x_{2J1})^t = R^{-1}(J_1)\ \text{Rect}_{J2}{}^{Y}[S_2^{-1}(\overline{J_2})]^t. \qquad (6.2.10)$$

Combining (6.2.9), (6.2.10) and similar formulas, will give (6.2.1a).

Fast Algorithm Theorem 6.2 shows a way to transform one linear system into another as stated in (6.2.1a). In the course of the transformation the column vector variables X and Y are rectangularized. At the same time, the coefficient matrix L is broken up into two small matrices: the premultiplying matrix R^{-1} and the postmultiplying matrix $(S^{-1})^t$. A necessary and sufficient condition for the transformation is $L = S \otimes R$.

The speed of computation can be further improved if R and S belong to the Sequence E. In this case, $R^{-1} = \hat{R}/J_1$ and $(S^{-1})^t = \tilde{S}/J_2$. Therefore

$$\mathrm{Rect}_{J2}{}^{X} = \hat{R}(J_1)\ \mathrm{Rect}_{J2}{}^{Y}\ \tilde{S}(J_2)/n, \quad n = J_1 J_2. \qquad (6.2.11)$$

Then solution of the 2-plex system turns out to be a multiplication of three matrices, and a division by n.

The total number of arithmetic operations is

$$J_1^2 J_2 + J_1 J_2^2 + J_1 J_2 \simeq J_1 J_2 (J_1 + J_2) \qquad \text{if } J_1, J_2 \text{ are large.} \qquad (6.2.12)$$

As we all know, Gauss Elimination is the most popular method for solving a system of equations with non-diagonally dominant coefficients. Gauss Elimination requires $J_1^3 J_2^3/3$ number of multiplications in this case. Comparison between these methods is illustrated in the following table:

Number of Complex Unknowns $n=J_1 \times J_2$	Estimated CPU Time on IBM 360/67 machine		
	(a) Gauss Elim.	(b) Fast Algor.	Ratio (a)/(b)
16	55 m-sec.	5 m-sec.	11
100	14 sec.	80 m-sec.	166
900	162 min.	2.2 sec.	4420

A Numerical Example Solve:

$$\begin{pmatrix} .3 & .3 & .35 & .35 & -.3 & -.3 & -.05 & -.05 \\ .3 & .6 & .35 & .7 & -.3 & -.6 & -.05 & -.1 \\ -.6 & -.6 & .3 & .3 & .6 & .6 & .1 & .1 \\ -.6 & -1.2 & .3 & .6 & .6 & 1.2 & .1 & .2 \\ -.1 & -.1 & .05 & .05 & .1 & .1 & -.15 & -.15 \\ -.1 & -.2 & .05 & .1 & .1 & .2 & -.15 & -.3 \\ .9 & .9 & -.45 & -.45 & .1 & .1 & .35 & .35 \\ .9 & 1.8 & -.45 & -.9 & .1 & .2 & .35 & .7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 1.35 \\ 2.3 \\ 1.3 \\ 1.4 \\ 0.05 \\ -.10 \\ -.45 \\ -.1 \end{pmatrix} \tag{6.2.13}$$

The coefficient matrix can be decomposed into a Kronecker product.

$$L = \begin{pmatrix} .3 & .35 & -.3 & -.05 \\ -.6 & .3 & .6 & .1 \\ -.1 & .05 & .1 & -.15 \\ .9 & -.45 & .1 & .35 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} = S \otimes R \tag{6.2.14}$$

By a usual method, we can find

$$S^{-t} = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 2 & 0 & 3 & -6 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad R^{-1} = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \tag{6.2.15}$$

According to Theorem 6.2, $\text{Rect}_4 X = R^{-1}\ \text{Rect}_4 Y\ S^{-t}$, therefore

$$\begin{pmatrix} x_1 & x_3 & x_5 & x_7 \\ x_2 & x_4 & x_6 & x_8 \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1.35 & 1.3 & 0.05 & -.45 \\ 2.3 & 1.4 & -.1 & -.1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 2 & 0 & 3 & -6 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix} . \qquad (6.2.16)$$

6.3 Solution of an m-plex System by Rectangularizing the Unknown Vector

The technique developed in the last section will be extended to an m-plex system $L(n)X(n') = Y(n')$, where $L(n)$ is a Kronecker product of m matrices:

$$L(J_1J_2J_3\ldots J_m) = S_m(J_m) \otimes S_{m-1}(J_{m-1}) \otimes \ldots \otimes S_2(J_2) \otimes S_1(J_1). \qquad (6.3.1)$$

Theorem 6.3.1 If $L(n) = S_r(J_r) \otimes S_{r-1}(J_{r-1})$, system $L\,X = Y$ can be transformed into

$$S_r(J_r)\ \mathrm{Rect}^t_{Jr}X = \mathrm{Rect}^t_{Jr}Y\ S^{-t}_{r-1}(J_{r-1}) . \qquad (6.3.2)$$

Proof: From (6.2.1) it is clear that the solution of

$$S_r \otimes S_{r-1}\ X = Y \qquad (6.3.3)$$

is

$$\mathrm{Rect}_{Jr}X = S^{-1}_{r-1}\ \mathrm{Rect}_{Jr}Y\ S^{-t}_r . \qquad (6.3.4)$$

Transposing both sides of (6.3.4)

$$\text{Rect}^t_{Jr} X = (S^{-1}_{r-1}\ \text{Rect}_{Jr} Y\ S^{-t}_r)^t = S^{-1}_r\ \text{Rect}^t_{Jr} Y\ S^{-t}_{r-1}. \quad (6.3.5)$$

Premultiplying (6.3.5) with S_r establishes (6.3.2).

Theorem 6.3.2 If the coefficient matrix of an m-plex system takes the form of (6.3.1), then the system can be transformed into

$$S_m\ \text{Rect}^t_{J(m-1)}\ \text{Rect}^t_{J(m-2)} \cdots \text{Rect}^t_{J2}\ \text{Rect}^t_{J1}\ X =$$

$$\text{Rect}^t_{J(m-1)} (\text{Rect}^t_{J(m-2)} (\ldots (\text{Rect}^t_{J1} Y S^{-t}_1) S^{-t}_2) \ldots.) \times$$

$$S^{-t}_{m-2}) S^{-t}_{m-1} \quad . \quad (6.3.7)$$

Proof: Let the system be

$$S_m \otimes S_{m-1} \otimes \ldots \otimes S_2 \otimes S_1\ X = Y \quad . \quad (6.3.8)$$

Theorem 6.3.1 will transform (6.3.8) into

$$S_m \otimes S_{m-1} \otimes \ldots \otimes S_2\ \text{Rect}^t_{J1} X = \text{Rect}^t_{J1} Y\ S^{-t}_1. \quad (6.3.9)$$

Again, Theorem 6.4.1 will turn (6.3.9) into

$$S_m \otimes S_{m-1} \otimes \ldots \otimes S_3\ \text{Rect}^t_{J2}\ \text{Rect}^t_{J1}\ X =$$

$$\text{Rect}^t_{J2} (\text{Rect}^t_{J1} Y S^{-t}_1) S^{-t}_2 \quad . \quad (6.3.10)$$

The process continued in this manner will yield (6.3.7).

(6.3.7) leads to the

Fast Algorithm The unique solution to the m-plex system

$$S_m \otimes S_{m-1} \otimes \ldots \otimes S_2 \otimes S_1 \, X = Y \qquad (6.3.11)$$

is

$$\mathrm{Rect}^t_{J(m-1)} \mathrm{Rect}^t_{J(m-2)} \mathrm{Rect}^t_{J(m-3)} \ldots R^t_{J2} \mathrm{Rect}^t_{J1} \, X =$$

$$S_m^{-1} \mathrm{Rect}^t_{J(m-1)} (\mathrm{Rect}^t_{J(m-2)} (\ldots (\mathrm{Rect}^t_{J2} (\mathrm{Rect}^t_{J1} \, Y \times$$

$$S_1^{-t}) S_2^{-t}) \ldots) S_{m-2}^{-t}) S_{m-1}^{-t} \, . \qquad (6.3.12)$$

6.4 Matrix Inversion by Rectangularization Techniques

Finding a matrix inverse by solving a system of equations is a well-known method. Now that we have a fast method to solve a system of equations, we hope that our fast method can help to invert a matrix.

Let us begin by replacing the data Y in (1.2.1) with a unit matrix I(n):

$$M(n) \, X(n) = I(n) \, . \qquad (6.4.1)$$

Solving this system of equations, we shall obtain X(n) which is the inverse of M(n):

$$X(n) = M^{-1}(n) \, I(n) = M^{-1}(n) \, . \qquad (6.4.2)$$

In a special case when $M(n) = L(n) = S(J_2) \otimes R(J_1)$, $n = J_1 J_2$, the system (6.4.1) can be solved by the fast method (6.2.1a), namely

$$\mathrm{Rect}_{J2}(L^{-1}(n)) = R^{-1}(J_1)(\mathrm{Rect}_{J2} I) S^{-t}(J_2) \, . \qquad (6.4.3)$$

Then we have the $L^{-1}(n)$ after rearranging the elements of $\text{Rect}_{J2}(L^{-1}(n))$. Since $\text{Rect}_{J2}I$ contains nothing but zeros and ones, it needs no arithmetic operations when multiplied by another matrix. The whole equation (6.4.3) needs $J_1^2 J_2^2$ multiplications. This is the same number of operations required for calculating $L^{-1}(n)$ using (5.2.4).

6.5 Summary

We provide a table to compare all the methods introduced in this chapter. This table does not include the necessary number of operations to invert R and S.

AIM	MATRIX INVERSION $L^{-1} = (S \otimes R)^{-1}$		
METHOD	Gauss.Elim.	Kronecker Mat. Inv.	Fast Algor.
FORMULA	L^{-1}	$S^{-1} \otimes R^{-1}$	Rect L^{-1} = R^{-1}(Rect I) S^{-t}
NO. OF MULTIP. REQ'D.	$J_1^3 J_2^3 - 1$	$J_1^2 J_2^2$	$J_1^2 J_2^2$
REF.	Westlake (1968)	(5.2.4)	(6.4.3)

AIM	SOLVING A SYSTEM OF EQUATIONS $LX=(S \otimes R)X=Y$		
METHOD	Gauss.Elim.	Kronecker Mat. Inv.	Fast Algor.
FORMULA	$X = L^{-1} Y$	$X=(S^{-1} \otimes R^{-1})Y$	$\text{Rect}\, X=R^{-1}(\text{Rect}\, Y)\, S^{-t}$
NO. OF MULTIP. REQ'D.	$J_1^3 J_2^3/3 + J_1^2 J_2^2 + J_1 J_2/3$	$2J_1^2 J_2^2$	$J_1 J_2 (J_1+J_2)$
REF.	Westlake (1968)	(6.1.2)	(6.2.12) (6.2.1a)

In the light of the above, we conclude that (6.2.1a) is the best algorithm for solving a system of equations with $L = S \otimes R$ as its coefficient matrix. However, if our aim is merely to find an inverse of L, then (5.2.4) and (6.4.3) are equally efficient.

PART II

SOLUTION TO OPTICAL

INVERSE SCATTERING PROBLEM

VII. OUR APPROACH TO A COMPUTATIONAL SOLUTION

7.1 The Physical Problem

7.1a Scalar Treatment of a Vector Field

A time-harmonic of an electromagnetic field has two components: an electric field and a magnetic field, each with three spatial components. Thus the electric field is written as $E \equiv (E_x, E_y, E_z)^t$ and the magnetic field as $H \equiv (H_x, H_y, H_z)^t$. It is our usual practice to study only the x-directional component of the electric field, which we call E_x. The other components will behave in a similar manner as E_x.

The component E_x is a function of time and space, with a frequency ω and a leading phase angle ϕ. Mathematically, it is

$$E_x \equiv E_x(x,y,z,t) \equiv A(x,y,z)\cos[\omega t+\phi(x,y,z)]$$

$$\equiv \mathrm{Re}[u(x,y,z)\exp(-i\omega t)] \ ,$$

where $u(x,y,z) = A(x,y,z)\exp(-i\phi)$ is the complex amplitude of E_x.

To study the optical field, it is sufficient to study u only.

7.1b Propagation in Isotropic Non-homogeneous Medium

An optical wave in vacuum satisfies the homogeneous Helmholtz equation. In the presence of a semi-transparent object with refractive index $n(x,y,z)$, which varies slowly in space such that $|(\nabla^2 n^2)/n^2| << (10)^{14}$ meter^{-2} (see Appendix V), the optical wave will be propagated according to

$$\nabla^2 u + k_o^2 n^2 u = 0 \quad . \tag{7.1.1}$$

Equation (7.1.1) is a Schrödinger's equation, where $k_o = 2\pi/\lambda$ is the wave number of a coherent monochromatic source of light, and ∇^2 is the Laplace operator.

7.1c The Scattered Field

In the space where light is scattered, the optical field is composed of two fields: the source-free background field and the field scattered by the object. Let us denote these fields with u_o and u_{sc}.

$$u = u_o + u_{sc} \quad . \tag{7.1.2}$$

Since u_o propagates in a homogeneous medium, it can be described by a Helmholtz equation:

$$\nabla^2 u_o + k_o^2 u_o = 0, \quad \text{or} \quad \nabla^2 u_o = - k_o^2 u_o \quad . \tag{7.1.3}$$

Substituting (7.1.2) and (7.1.3) into (7.1.1) yields

$$\nabla^2 u_{sc} + k_o^2 u_{sc} = -k_o^2 (n^2-1)\, u \,. \qquad (7.1.4)$$

Let us define the scattering potential $f_s \equiv -k_o^2(n^2 - 1)$. Then

$$\nabla^2 u_{sc} + k_o^2 u_{sc} = f_s\, u \,. \qquad (7.1.5)$$

7.1d Born's First Approximation

In many cases, the field u_{sc} scattered by a semi-transparent object is weak compared to the background field u_o. To simplify (7.1.5) we shall, according to Born's theory, replace u with u_o so that the right hand side of (7.1.5) becomes independent of u_{sc}. Then the equation for the propagation of u_{sc}

$$\nabla^2 u_{sc} + k_o^2 u_{sc} = f_s\, u_o \qquad (7.1.6)$$

is a non-homogeneous Helmholtz partial differential equation.

A particular solution of (7.1.6) is (see Appendix VI)

$$u_{sc} = \int_V G\, f_s\, u_o\, dV \qquad (7.1.7)$$

with V representing the volume of the scattering object, $dV \equiv dx\, dy\, dz$ (see Fig.1),

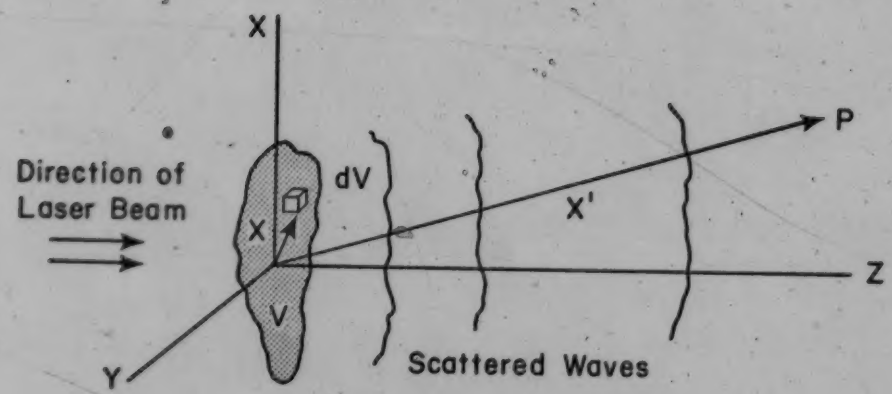


FIG. 1 THE SPACE VECTORS

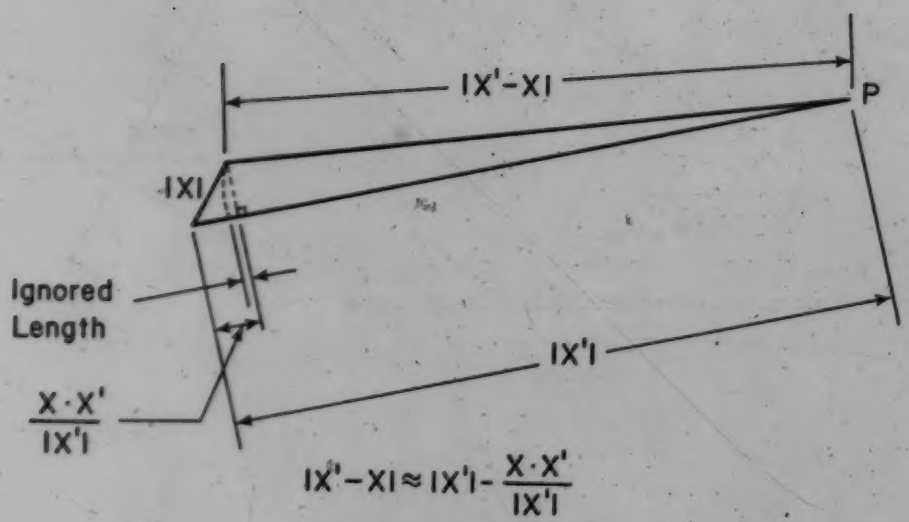


FIG. 2 THE FAR-FIELD APPROXIMATION

G denoting the Green's function given by

$$G = \frac{-\exp(i\, k_o |X'-X|)}{4\pi |X' - X|} , \quad \text{with } i^2 = -1 \qquad (7.1.8)$$

$X \equiv [x,y,z]^t$ being a space vector pointing to the position of dV inside the scattering object,

$X' \equiv [x',y',z']^t$ being another vector pointing to a point (x',y',z') either in the far-field region where u_{sc} is measured, or within the scatterer.

7.1e The Far-Field Approximation

Subject to the condition (see Schmidt-Weinmar, 1971)

$$|X|^2/\lambda << |X'| , \qquad (7.1.9)$$

$|X'- X|$ can be approximated by $|X'| - \frac{X \cdot X}{|X'|}$ for the numerator in (7.1.8) and by $|X'|$ for the denominator. The former approximation is illustrated in Figure 2. Here "·" indicates an inner vector product.

We shall introduce a new vector K whose magnitude is always a constant equal to k_o, and whose direction will be the same as the far-field pointer X':

$$K \equiv k_o \frac{X'}{|X'|} = (K_x, K_y, K_z)^t . \qquad (7.1.10)$$

Then (7.1.7) is simplified to

$$u_{sc} = \frac{\exp(ik_o|X'|)}{4\pi|X'|} \int_V \exp(-i\ K\cdot X)(-f_s u_o)\ dV. \qquad (7.1.11)$$

To convert the above to a digitized form, we let

$$F \equiv \frac{4\pi|X'|u_{sc}}{\exp(ik_o|X'|)\phi(K)\Delta V}\ , \qquad (7.1.12)$$

where $\phi(K)$ is a form factor, and $\Delta V = \Delta x\Delta y\Delta z$ is the digital equivalent of dV. We also let

$$f \equiv -\ f_s u_o\ . \qquad (7.1.13)$$

Appendix IV gives details of the mathematical derivation. A result is yielded as

$$F = \sum_x \sum_y \sum_z \exp(-iK\cdot X)\ f\ . \qquad (7.1.14)$$

It should be noted that f represents the source densities due to the multiple scattering sources within a volume element ΔV that has its center at x. In (7.1.13) f is a product of u_o and f_s, both being functions of x. Hence f depends on x. f does not depend on K.

K is a vector of constant length k_o, and aligned in the direction of X' (see 7.1.10). As the point P moves about in the far-field region, K sweeps a spherical surface. Any point on this sphere, as represented

by K, has only two degrees of freedom instead of three. If K_x and K_y are the two free dimensions, then the third dimension is determined by K_x and K_y. Hence $K = (K_x, K_y, K_z)^t$; $K_z = \sqrt{K_o^2 - K_x^2 - K_y^2}$.

F is proportional to u_{sc} (see 7.1.12). It represents the scattered field in the far-field region. The magnitude and phase of u_{sc} can be measured under far-field conditions (Schmidt-Weinmar, 1975(1)) and hence F may now be considered as a set of known quantities, which are the input data to our inverse scattering problem. Since u_{sc} is a function of X', which is a function of K, we conclude that F has two degrees of freedom represented by K_x and K_y.

7.2 Discrete Representations

The rest of this chapter will be devoted to defining our problem in terms amenable to numerical analysis. This section describes a discrete coordinate system in the space domain. The next section will establish a system of equations suitable for a computer solution.

At the moment, we are dealing with a semi-transparent weakly scattering object of a finite size. We shall assume that it is contained in a rectangular block-shaped space. The space is subdivided into

rectangular blocklets of equal sizes. The scattered field source within each blocklet will be averaged and assumed to be originated from the blocklet center.

Our spatial coordinates are chosen such that the entire scatterer lies in the first octant, and axes are parallel to the edges of the block. If we call the dimensions of each blocklet Δx, Δy, Δz, then the coordinate origin should be placed at a distance $[\Delta x, \Delta y, \Delta z]^t$ from the center of a corner blocklet (Fig. 3).

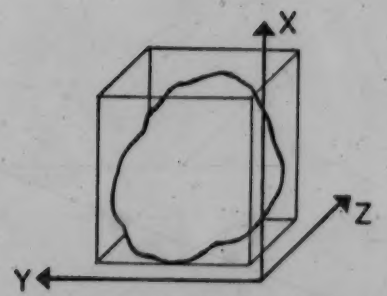
Let J_1, J_2, J_3 be the number of blocklets in the x-, y-, z-directions. Let also j_1, j_2, j_3 be the count numbers in the three directions. Let again $X \equiv [x\ y\ z]^t$ be the coordinates of the centers of the blocklets. Then our choice of coordinate system above results in

$$\begin{aligned} x &\equiv j_1 \Delta x \qquad j_1 \equiv \{1,2,\ldots,j_{1v},\ldots,J_1\} \\ y &\equiv j_2 \Delta y \qquad j_2 \equiv \{1,2,\ldots,j_{2v},\ldots,J_2\} \\ z &\equiv j_3 \Delta z \qquad j_3 \equiv \{1,2,\ldots,j_{3v},\ldots,J_3\} \end{aligned} \tag{7.2.1}$$

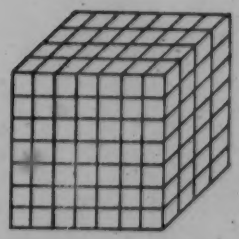
For convenience in writing, we shall label the blocklets with an integer v where

$$v \equiv j_{1v} + J_1(j_{2v}-1) + J_1J_2(j_{3v}-1) \ . \tag{7.2.2}$$

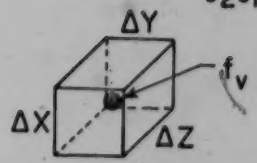
(7.2.2) shows a one-to-one mapping between the count numbers j_{1v}, j_{2v}, j_{3v} and the new label v. Obviously,



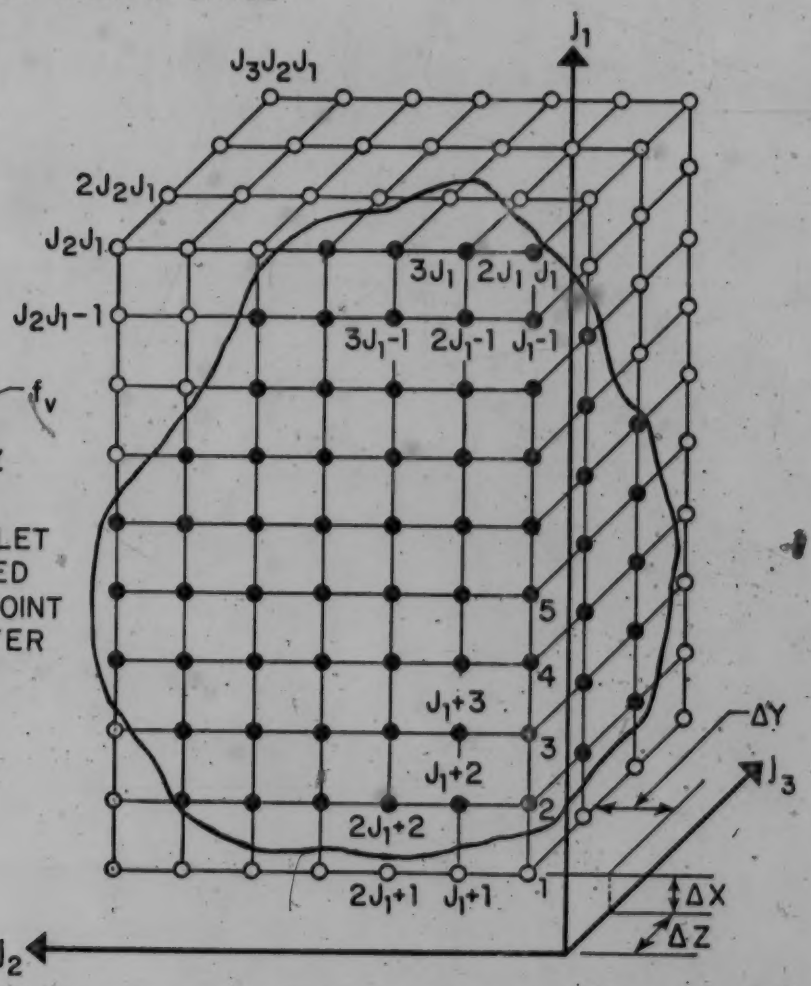
A SCATTERER AND
ITS OCCUPIED
BLOCK-SHAPED SPACE



THE SPACE IS SUBDIVIDED
INTO BLOCKLETS



THE v th BLOCKLET
IS REPRESENTED
BY A SOURCE POINT
f_v AT ITS CENTER
(x_v, y_v, z_v)



THE SCATTERER IS REPRESENTED BY $J_1 J_2 J_3$ SOURCE POINTS
(The numbers attached to source points are the values of v)

FIG. 3

from (7.2.1) and (7.2.2), $v = \{1,2,\ldots,J_1J_2J_3\}$.

All variables associated with a blocklet can now be labelled. The first variable involved is X. Attaching v to X and modifying (7.2.2),

$$X_v \equiv [x_v, y_v, z_v]^t = [j_{1v}\Delta x,\ j_{2v}\Delta y,\ j_{3v}\Delta z]^t. \qquad (7.2.3)$$

The second variable associated with blocklets is f. f is a hypothetical quantity, located at the center of each blocklet (see Fig. 3), to represent a mean value of scattered-field source density within the blocklet. The definition of f is given by equation (7.1.13) as the product of the scattering potential and the background field. f is a function of X_v. We shall attach the same subscript v to X and f:

$$f = f_v \quad \text{when} \quad X = X_v . \qquad (7.2.4)$$

Since all the f are subscripted, these f_v, for $v = 1,2,\ldots,J_1J_2J_3$, will be listed together to form a vector f(n') with $n = J_1J_2J_3$.

$$f(n') = [f_1\ f_2\ \cdots\ f_{J1J2J3}]^t \qquad (7.2.5)$$

Now the space domain has been discretized to form a uniformly spaced grid structure. The grid points are

finite in number, labelled from 1 to $J_1J_2J_3$, and located at X_v where X_v is defined in (7.2.3). The scatterer is represented by the grid structure in which to each grid point we attach a hypothetical quantity called the scattered-field source density f_v which is defined in (7.1.13).

7.3 The System of Equations

Each point on the grid structure, except those points with $f_v = 0$, is considered the location of an independent source of optical disturbance. It emits an optical wave which propagates in a manner described by equation (7.1.1). The combined effect of all these sources creates the actual optical field in the far-field region.

The scattered far field is represented by F. According to equation (7.1.14), F is related to f_v by

$$F = \sum_v \exp(-iK\cdot X_v)\ f_v \ . \qquad (7.3.2)$$

In our inverse scattering problem, we want to determine the unknowns $f_1, f_2, \ldots, f_{J1J2J3}$ from the data F. It is necessary to select $J_1J_2J_3$ independent

equations to solve for these unknowns simultaneously. How to select these equations to expedite our numerical computation is a topic to be dealt with in Chapter IX. At this stage it may be sufficient to assume that we can choose, in some suitable way, $J_1J_2J_3$ vectors K, labelled as K_u with

$$u = 1,2,\ldots,J_1J_2J_3 \ . \tag{7.3.3}$$

The sets of data F should correspond to K_u, and be labelled as F_u:

$$F = F_u \quad \text{when} \quad K = K_u \ . \tag{7.3.4}$$

Arranging the F_u in the order of u, we have the vector F(n').

$$F(n') \equiv [F_1 \ F_2 \ \ldots \ F_{J1J2J3}]^t \ . \tag{7.3.5}$$

Then the selection of K_u yields a system of equations

$$\begin{cases} F_1 = \sum\limits_v \exp(-iK_1 \cdot X_v) \ f_v \\ F_2 = \sum\limits_v \exp(-iK_2 \cdot X_v) \ f_v \\ \quad \ldots\ldots \\ F_{J1J2J3} = \sum\limits_v \exp(-iK_{J1J2J3} \cdot X_v) \ f_v \end{cases} \tag{7.3.6}$$

Writing (7.3.6) in matrix form with the help of (7.3.5) and (7.2.5),

$$F(n') = P(n)f(n') \tag{7.3.7}$$

where

$$P(n) = \begin{bmatrix} \exp(-iK_1 \cdot X_1) & \exp(-iK_1 \cdot X_2) & \cdots & \exp(-iK_1 \cdot X_{J1J2J3}) \\ \exp(-iK_2 \cdot X_1) & \exp(-iK_2 \cdot X_2) & \cdots & \exp(-iK_2 \cdot X_{J1J2J3}) \\ & \cdots\cdots\cdots\cdots\cdots & & \\ \exp(-iK_{J1J2J3} \cdot X_1) & \cdots & & \exp(-iK_{J1J2J3} \cdot X_{J1J2J3}) \end{bmatrix} \tag{7.3.8}$$

Let the (u,v)th element of P be p_{uv}; then

$$p_{uv} \equiv \exp(-iK_u \cdot X_v) \ . \tag{7.3.9}$$

Since P is not diagonally dominant, solution of (7.3.6) by iterative methods will not converge. An attempt was made to solve (7.3.6) by Gaussian Elimination; the computing time taken on IBM/360 was:

No. of Complex Unknowns	Required CPU time in seconds
27	3
64	37
400	533
2000	86400

For large number of complex unknowns, solving (7.3.7) by Gaussian Elimination should be impracticable.

VIII. METHOD OF SOLUTION

8.1 Outline of the Method

Our method in solving $Pf = F$ is guided by the following:

1. The values of $K_u = [K_{xu}, K_{yu}, K_{zu}]^t$ for $u = 1,2,\ldots,n$ will be taken to be n different points on a sphere in the K-domain [see equation (7.1.10)]. This spherical surface is continuous. We can choose any n points on this spherical surface as $K_1, K_2, \ldots, K_n$.
2. The selection of K_u will not affect the unknowns f_v, because f_v is a function of X_v, not of K_u (see Section 7.1).
3. The selection of K_u will affect p_{uv} and F_u (see 7.3.9 and 7.3.2). We hope that a clever selection of K_u may simplify P to one of our matrices in Part I. If this can be achieved then we shall apply the fast algorithms that we have introduced in Chapter II through Chapter VI.

As we noticed in (7.3.9), every element of P involves an inner vector product $K_u \cdot X_v$, which is a sum of three terms.

$$K_u \cdot X_v = K_{xu} x_v + K_{yu} y_v + K_{zu} z_v \,. \qquad (8.1.1)$$

It would be difficult to identify P with the patterns of any simple sequence that we previously introduced. The best we can hope for is that by a selection of K_u, P can be turned into a product of some sort involving three factor matrices, so that the system Pf = F will be solved eventually by matrix decomposition techniques.

Our method is based on treating P as a row-wise Kronecker product sequence W, consisting of a Kronecker sequence L and a Vandermonde sequence C. The matrices in sequence C can readily be inverted as a partial solution. Then the rest of the system will be handled as a 2-plex system with L = R ⊗ S as its coefficient matrix, and R and S will be inverted as another two partial solutions. All these partial solutions are then pieced together to yield a final answer.

The following sections give a detailed explanation.

8.2 Creation of Row Redundancies

Let us rewrite (7.3.9) in a form more convenient for manipulation. By (8.1.1) and (7.2.3), with Int{A} indicating the integer part of a real number A, and (B) mod(C) indicating the remainder integer resulting from B/C.

$$P_{uv} = \exp(-iK_u \cdot X_v) = \exp\{-i(x_v K_{xu} + y_v K_{yu} + z_v K_{zu})\}$$

$$= \exp\{-i(j_1 \Delta x K_{xu} + j_2 \Delta y K_{yu} + j_3 \Delta z K_{zu})\}$$

$$= r_u'^{j1}\, s_u'^{j2}\, t_u^{j3} , \qquad (8.2.1)$$

where

$$v = 1, 2, \ldots, J_1 J_2 J_3 ,$$

$$j_1 = \{(v-1) \bmod (J_1)\} + 1 ;$$

$$j_2 = \{\mathrm{Int}[(v-1)/J_3]\} \bmod (J_2) + 1 ;$$

$$j_3 = \mathrm{Int}\{(v-1)/J_2 J_3\} + 1 ; \qquad (8.2.2)$$

$$r_u' \equiv \exp(-i \Delta x\, K_{xu}) ;$$

$$s_u' \equiv \exp(-i \Delta y\, K_{yu}) ;$$

$$t_u \equiv \exp(-i \Delta z\, K_{zu}) .$$

The relations for j_1, j_2, j_3 here are the inverse functions of (7.2.2).

j_1, j_2, j_3 will be increased lexicographically according to (7.2.1). We shall obtain a row $P_u(\bar{n})$ from (8.2.1) for a particular value of u, and in this row, $J_2 J_3$ elements of P will have the same factor $r_u'^{j1}$, $J_3 J_1$ elements will have the same factor $s_u'^{j2}$, and $J_1 J_2$ elements will have the same factor t_u^{j3}.

8.3 Creation of Column Redundancies

In order to turn P(n) into W(n), only redundancy in the rows of P(n) is not enough. We must create column-wise redundancy (see Section 5.3).

Column-wise redundancy can be obtained by selecting K_u in special ways, and there are two possible patterns of redundancies. These will be discussed in detail in Chapter XI. In this chapter, let us assume that we can succeed in splitting the n variables r'_u into J_1J_2 groups, and assign one common value to the J_3 variables r'_u in each group. This means

$$r'_u = r_{k1} = \text{constant}$$

$$k_1 = \{\text{Int}[(u-1)/J_3]\}\text{mod}(J_1) + 1 = 1,2,\ldots,J_1 . \qquad (8.3.1)$$

In a similar manner, we shall assume that we can achieve

$$s'_u = s_{k2} = \text{constant}$$

$$k_2 \equiv \text{Int}\{(u-1)/J_3J_1\} + 1 = 1,2,\ldots,J_2 . \qquad (8.3.2)$$

(8.3.1) and (8.3.2) change (8.2.1) into

$$P_{uv} = r_{k1}^{j1}\, s_{k2}^{j2}\, t_u^{j3} . \qquad (8.3.3)$$

We did not assign common values to the variable t_u. Nor did we change the subscript u for t. This is

due to the fact that only two degrees of freedom K_x and K_y were assigned to K. K_z will be dependent on K_x and K_y (see Section 7.1). Therefore t_u cannot be assigned a common value but depends on the values chosen for r'_u and s'_u.

(8.3.3) yields the following format for P(n):

$$P(n) = \begin{vmatrix} r_1 s_1 t_1 & r_1^2 s_1 t_1 & r_1^3 s_1 t_1 \ldots & r_1^{J1} s_1 t_1 & r_1 s_1^2 t_1 \ldots & r_1^{J1} s_1^{J2} t_1^{J3} \\ r_1 s_1 t_2 & r_1^2 s_1 t_2 & r_1^3 s_1 t_2 & \ldots\ldots & & r_1^{J1} s_1^{J2} t_2^{J3} \\ & \ldots\ldots\ldots & & & & \ldots\ldots \\ r_1 s_1 t_{J3} & \ldots\ldots\ldots & & & & r_1^{J1} s_1^{J2} t_{J3}^{J3} \\ r_2 s_1 t_{J3+1} & \ldots\ldots\ldots & & & & r_2^{J1} s_1^{J2} t_{J3+1}^{J3} \\ r_2 s_1 t_{J3+2} & \ldots\ldots\ldots & & & & \ldots\ldots \\ \ldots\ldots & & & & & \\ \ldots\ldots & & & & & \ldots\ldots \\ r_{J1} s_1 t_{J2J3} & \ldots\ldots\ldots & & & & \\ r_1 s_2 t_{J2J3+1} & \ldots\ldots\ldots & & & & \ldots\ldots \\ \ldots\ldots & & & & & \\ \ldots\ldots & & & & & \\ r_{J1} s_{J2} t_{J1J2J3} & & & \ldots\ldots\ldots & & r_{J1}^{J1} s_{J2}^{J2} t_n^{J3} \end{vmatrix}$$

(8.3.4)

8.4 Algebraic Solution

The matrix P(n) has $J_1 J_2 J_3$ rows. The uth row is

$$
\begin{aligned}
P_u(n) = [& r'_u s'_u t_u \; r'^2_u s'_u t_u \; r'^3_u s'_u t_u \; \ldots\ldots \; r'^{J1}_u s'_u t_u \\
& r'_u s'^2_u t_u \; r'^2_u s'^2_u t_u \; r'^3_u s'^2_u t_u \; \ldots\ldots \; r'^{J1}_u s'^2_u t_u \\
& \ldots\ldots\ldots\ldots\ldots\ldots \\
& r'_u s'^{J2}_u t_u \; r'^2_u s'^{J2}_u t_u \; r'^3_u s'^{J2}_u t_u \ldots\ldots r'^{J1}_u s'^{J2}_u t_u \\
& r'_u s'_u t^2_u \; r'^2_u s'_u t^2_u \; r'^3_u s'_u t^2_u \; \ldots\ldots \; r'^{J1}_u s'_u t^2_u \\
& r'_u s'^2_u t^2_u \; r'^2_u s'^2_u t^2_u \; r'^3_u s'^2_u t^2_u \; \ldots\ldots \; r'^{J1}_u s'^2_u t^2_u \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; r^{J1}_u s^{J2}_u t^2_u \\
& r'_u s'_u t^3_u \; r'^2_u s'_u t^3_u \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
& \ldots\ldots\ldots\ldots\ldots\ldots \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; r'^{J1}_u s'^{J2}_u t^{J3}_u].
\end{aligned}
\tag{8.4.1}
$$

Letting $u = 1,2,3,\ldots,J_1J_2J_3$, the whole matrix P can be written out.

Subject to the format (8.3.4), P(n) can be re-written as

$$
P(n) = \begin{bmatrix}
{}_1T(J_3) \otimes L_1(\overline{J_1J_2}) \\
{}_2T(J_3) \otimes L_2(\overline{J_1J_2}) \\
\ldots\ldots\ldots \\
{}_{J1J2}T(J_3) \otimes L_{J1J2}(\overline{J_1J_2})
\end{bmatrix}
\tag{8.4.2}
$$

where

$$
\left.
\begin{aligned}
{}_1T(J_3) &= \begin{pmatrix} t_1 & t_1^2 & t_1^3 & \cdots & t_1^{J3} \\ t_2 & t_2^2 & t_2^3 & \cdots & t_2^{J3} \\ t_3 & \cdots & \cdots & & t_3^{J3} \\ \cdots & \cdots & & & \\ t_{J3} & \cdots & \cdots & & t_{J3}^{J3} \end{pmatrix} \\
L_1(\overline{J_1J_2}) &= [r_1s_1 \;\; r_1^2s_1 \;\; r_1^3s_1 \;\; \cdots \;\; r_1^{J1}s_1^{J2}] \\
{}_2T(J_3) &= \begin{pmatrix} t_{J3+1} & t_{J3+1}^2 & \cdots & t_{J3+1}^{J3} \\ t_{J3+2} & t_{J3+2}^2 & \cdots & t_{J3+2}^{J3} \\ \cdots & \cdots & & \\ t_{2J3} & t_{2J3}^2 & \cdots & t_{2J3}^{J3} \end{pmatrix} \\
L_2(\overline{J_1J_2}) &= [r_2s_1 \;\; r_2^2s_1 \;\; \cdots \;\; r_2^{J1}s_1^{J2}] \\
&\cdots\cdots\cdots\cdots\cdots \\
{}_{J1J2}T(J_3) &= \begin{pmatrix} t_{(J1J2-1)J3+1} & \cdots & t_{(J1J2-1)J3+1}^{J3} \\ \cdots & \cdots & \\ t_{J1J2} & t_{J1J2}^2 & \cdots & t_{J1J2}^{J3} \end{pmatrix} \\
L_{J1J2}(\overline{J_1J_2}) &= [r_{J1}s_{J2} \;\; r_{J1}^2s_{J2} \;\; \cdots \;\; r_{J1}^{J1}s_{J2}^{J2}] \;.
\end{aligned}
\right\} \quad (8.4.3)
$$

(8.4.2) has the structure of the matrix W in (5.3.2).

The linear system of equations given by the matrix (8.4.2) consists of J_1J_2 sets of systems L of the form

$$ {}_m T(J_3) \otimes L_m(\overline{J_1 J_2}) f = \begin{pmatrix} F_{(m-1)J3+1} \\ F_{(m-1)J3+2} \\ \cdots\cdots \\ F_{(m-1)J3+J3} \end{pmatrix}, m = 1,2,\ldots,J_1 J_2. \quad (8.4.4) $$

According to (6.2.1) the above system can be transformed to

$$ L_m(\overline{J_1 J_2}) \mathrm{Rect}_{J3} f = \mathrm{Rect}_{J3} \begin{pmatrix} F_{(m-1)J3+1} \\ F_{(m-1)J3+2} \\ \cdots\cdots \\ F_{(m-1)J3+J3} \end{pmatrix} {}_m T^{-t}(J_3). \quad (8.4.5) $$

Letting $m = 1,2,\ldots,J_1 J_2$, (8.4.5) becomes

$$ \left.\begin{aligned} L_1(\overline{J_1 J_2}) \mathrm{Rect}_{J3} f &= \mathrm{Rect}_{J3}(F_1\ F_2\ \cdots\ F_{J3})\ {}_1 T^{-t}(J_3) \\ &= (F_1\ F_2\ \cdots\ F_{J3})\ {}_1 T^{-t}(J3) \\ L_2(\overline{J_1 J_2}) \mathrm{Rect}_{J3} f &= (F_{J3+1}\ F_{J3+2}\ \cdots\ F_{2J3})\ {}_2 T^{-t}(J_3) \\ &\cdots\cdots\cdots\cdots\cdots \\ L_{J1J2}(\overline{J_1 J_2}) \mathrm{Rect}_{J3} f &= (F_{(J1J2-1)J3+1} \cdots \\ &\qquad F_{J1J2J3})\ {}_{J1J2} T^{-t}(J_3)\ . \end{aligned}\right\} \quad (8.4.6) $$

The equations in (8.4.6) are combined to yield

$$ L(J_1 J_2) \mathrm{Rect}_{J3} f = \begin{pmatrix} (F_1 \cdots F_{J3})\ {}_1 T^{-t}(J_3) \\ (F_{J3+1} \cdots F_{2J3})\ {}_2 T^{-t}(J_3) \\ \cdots\cdots\cdots\cdots\cdots \\ (F_{(J1J2-1)J3+1} \cdots F_{J1J2J3})\ {}_{J1J2} T^{-t}(J_3) \end{pmatrix} \quad (8.4.7) $$

Let F" be the right hand side of (8.4.7).

$$L(J_1J_2)\ \mathrm{Rect}_{J3} f = F''(J_1J_2,J_3)\ . \qquad (8.4.8)$$

The structure of each row of the matrix $L(J_1J_2)$ is seen in (8.4.1) and (8.4.3). It indicates that L is a Kronecker matrix of the form

$$L(J_1J_2) = R(J_1) \otimes S(J_2)$$

where

$$R(J_1) = \begin{pmatrix} r_1 & r_1^2 & r_1^3 & \cdots & r_1^{J1} \\ r_2 & r_2^2 & r_2^3 & \cdots & r_2^{J1} \\ & \cdots\cdots & & & \\ r_{J1} & r_{J1}^2 & r_{J1}^3 & \cdots & r_{J1}^{J1} \end{pmatrix} \qquad (8.4.9)$$

$$S(J_2) = \begin{pmatrix} s_1 & s_1^2 & s_1^3 & \cdots & s_1^{J2} \\ s_2 & s_2^2 & s_2^3 & \cdots & s_2^{J2} \\ & \cdots\cdots & & & \\ s_{J2} & s_{J2}^2 & s_{J2}^3 & \cdots & s_{J2}^{J2} \end{pmatrix}$$

By applying Theorem 6.2, (8.4.8) will be transformed into

$$\mathrm{Rect}_{J2}\mathrm{Rect}_{J3} f = R^{-1}(J_1)\ \mathrm{Rect}_{J2} F''\ S^{-t}(J_2). \qquad (8.4.10)$$

This is our fast solution of $P(n)f(n') = F(n')$.

8.5 Computing Time

The approximate number of complex multiplications required for our fast algorithm is listed below:

Procedure	Main Actions Required	No. of Complex Multiplications
1. Input data	Read R,S,T,F. Find R^{-1}, S^{-1}.	Nil (see Section 9.5)
2. Invert J_1J_2 Vandermonde matrices $_mT(J_3)$.	Parker's method and Traub's algorithm I.	$7J_1J_2(J_3)^2/2$
3. Compute F"	Multiply $F_m(\overline{J_3})$ with $_mT^{-t}(J_3)$ for $m = 1,\ldots,J_1J_2$	$(J_1J_2)(J_3)^2$
4. Compute f	Multiply three matrices of sizes $J_1 \times J_1$, $J_1 \times J_2$, $J_2 \times J_2$ and divide result by n.	$J_1^2J_2+J_1J_2^2+J_1J_2J_3 =$ $(J_1J_2)(J_1+J_2+J_3)$

The above sum up to $J_1J_2(4.5J_3^2+J_1+J_2+J_3)$ complex multiplications.

Let $J_1 = J_2 = J_3 = 20$; then we have a system of 8000 equations to solve.

Using our fast algorithm, about 744000 multiplications are needed. If IBM 360/67 machines are used for computation, solution of the system of equations

needs only a minute execution time. (Compiling time is not included in the figure as the program and subroutines can be compiled beforehand and the object program is stored).

8.6 Storage Space

To store a coefficient matrix $P(J_1J_2J_3)$ would require $(J_1J_2J_3)^2$ complex words, whereas storing its Kronecker factor matrices R, S, and ${}_mT$ only needs $J_1^2 + J_2^2 + J_1J_2J_3^2$ complex words. As a matter of fact, we need not store the whole Vandermonde matrix. We can store only the first columns of R, S, T; then the other columns can be generated from the elements of their first column. In so doing we can reduce the storage space to $J_1 + J_2 + J_1J_2J_3$ complex words only. In the latter case, to generate the other columns implies extra multiplications. Whether this is worthwhile depends on the size of ${}_mT(J_3)$. If J_3 is big, the latter is preferable.

Let us assume $J_1 = J_2 = J_3 = 20$; storing the matrix P normally requires 64m complex words. Storing R, S; T in full needs 160K complex words only. Our algorithm does not require the matrix P, hence 160K complex words are sufficient for our use. If we store the first columns of R, S and T then it needs 8.4K complex words, with an extra $J_1J_2J_3^2/2$ multiplications to

generate the full matrices R, S, and T. This extra number in this case is 80000 multiplications, about 10% of the 744,000 multiplications originally required by our fast algorithm.

Furthermore, our algorithm handles the J_1J_2 matrices ${}_mT$, one at a time. A large number of work spaces can be assigned to the same core space. Thus it reduces the amount of core storage considerably.

IX. SELECTION OF PHASE ANGLES

We showed in Chapter VIII that a fast algorithm can be developed if we make some of the r'_u take a common value r_{k1} and also make some of the s'_u take a common value s_{k2}. We have, however, left out 2 important points unexplained in Section 8.3: How can we make r'_u and s'_u take common values? What is the physical meaning of equalizing some of these parameters? The current chapter will answer these questions in detail.

9.1 Principles of Selection

The elements of P are given by (7.3.9):

$$p_{uv} = \exp(-iK_u \cdot X_v) \ . \tag{9.1.1}$$

$K_u \cdot X_v$ may be considered as a phase angle of p_{uv}. The phase angle is resolved into 3 phase angle components described by (8.1.1), namely,

$$K_u \cdot X_v = j_1 \Delta x K_{xu} + j_2 \Delta y K_{yu} + j_3 \Delta z K_{zu} \ . \tag{9.1.2}$$

Then, p_{uv} is decomposed into 3 components in (8.2.1):

$$p_{uv} = r'^{j_1}_u \, s'^{j_2}_u \, t^{j_3}_u \ . \tag{9.1.3}$$

The relations between the p_{uv} components and phase angle components are stated in (8.2.2):

$$\begin{cases} r'_u = \exp(-i\Delta x K_{xu}) \\ s'_u = \exp(-i\Delta y K_{yu}) \\ t_u = \exp(-i\Delta z K_{zu}) \end{cases} \qquad (9.1.4)$$

In order to have repeated values for r'_u and s'_u, which are periodic functions, it is only necessary to select values modulo 2π of $\Delta x K_{xu}$ and $\Delta y K_{yu}$ for different values of u.

Now we have our first guiding principle for selecting the phase angles:

We must create redundancies by selecting values modulo 2π for $\Delta x K_{xu}$ and $\Delta y K_{yu}$.

We note that selecting a common value for r'_u may result in a singular matrix R (see 8.4.9) which contains only r_{kl} as its elements. Similar problems may also arise when we equalize the s'_u values. Furthermore, the selection of $\Delta x K_{xu}$ and $\Delta y K_{yu}$ will automatically produce the $\Delta z K_{zu}$ values (see Section 9.2). $\Delta z K_{zu}$ will, in turn, determine the values of t_u, and hence the matrices ${}_mT$, which may be singular as well. Therefore our second principle is:

The selection of phase components $\Delta x K_{xu}$ and $\Delta y K_{yu}$ should not result in any of R, S, ${}_mT$, or P matrices being singular, otherwise inversion of the matrix

concerned will be infeasible and our fast algorithm will fail.

Besides these principles, we have our third principle, namely, to select the phase angles so as to stabilize the Vandermonde matrices ${}_{m}T$. Our fourth principle is to select phase angles to turn matrices R and S into E sequences, etc. All these principles will be explained in the subsequent sections of this chapter.

9.2 Graphical Representation of Phase Angles

Let it be remembered that K_u is a three-dimensional vector with only two independent components as it is chosen on the surface of a sphere (Section 7.1 refers). If we select K_{xu} and K_{yu} for a particular u, then K_{zu} is determined by the equation of the sphere. Since Δx, Δy, Δz are chosen by our physical problem and we are not free to assign a value just for the sake of a fast algorithm, we can only select two of the three phase components of p_{uv}: Usually we select $\Delta x K_{xu}$ and $\Delta y K_{yu}$ and leave $\Delta z K_{zu}$ to be determined later by the selected values of $\Delta x K_{xu}$, $\Delta y K_{yu}$ and Δz.

In other words, we can now simplify our selection of phase angles to the selection of points in the plane with axes $\Delta x K_x$ and $\Delta y K_y$. For $u = 1,2,\ldots,n$, we are

required to select n points (ΔxK_{xu}, ΔyK_{yu}) in this plane. These points are actually the projections of n 3-dimensional phase angle vectors (ΔxK_{xu}, ΔyK_{yu}, ΔzK_{zu}) onto the (ΔxK_x, ΔyK_y) plane.

If ΔxK_{xu} and ΔyK_{yu} are subject to no restrictions, the n points or phase angles may be randomly scattered on the (ΔxK_x, ΔyK_y) plane as shown in Fig. 4. In a real-imaginary graphical representation of complex numbers, r'_u and s'_u terminate on a unit circle. r'_u and s'_u take phase angles of arbitrary magnitude as illustrated in the bottom picture of Fig. 4.

In order that P be turned into (8.3.4) we shall put restrictions on ΔxK_{xu} and ΔyK_{yu}. Let us recall what is required by (8.3.4):

(1) The $J_1J_2J_3$ rows of matrix P will be divided into J_1J_2 groups of J_3 rows each. The values of r'_u for every group takes a common value r_{k1} (see 8.3.1).

(2) The values of s'_u for every consecutive J_1 groups of J_3 rows each, are equal to a common value s_{k2} (see 8.3.2).

In addition to these, we have

(3) The values of t_u may be repeated for some rows. But t_u must not be repeated within any group of J_3 rows; otherwise ${}_mT(J_3)$ will have two

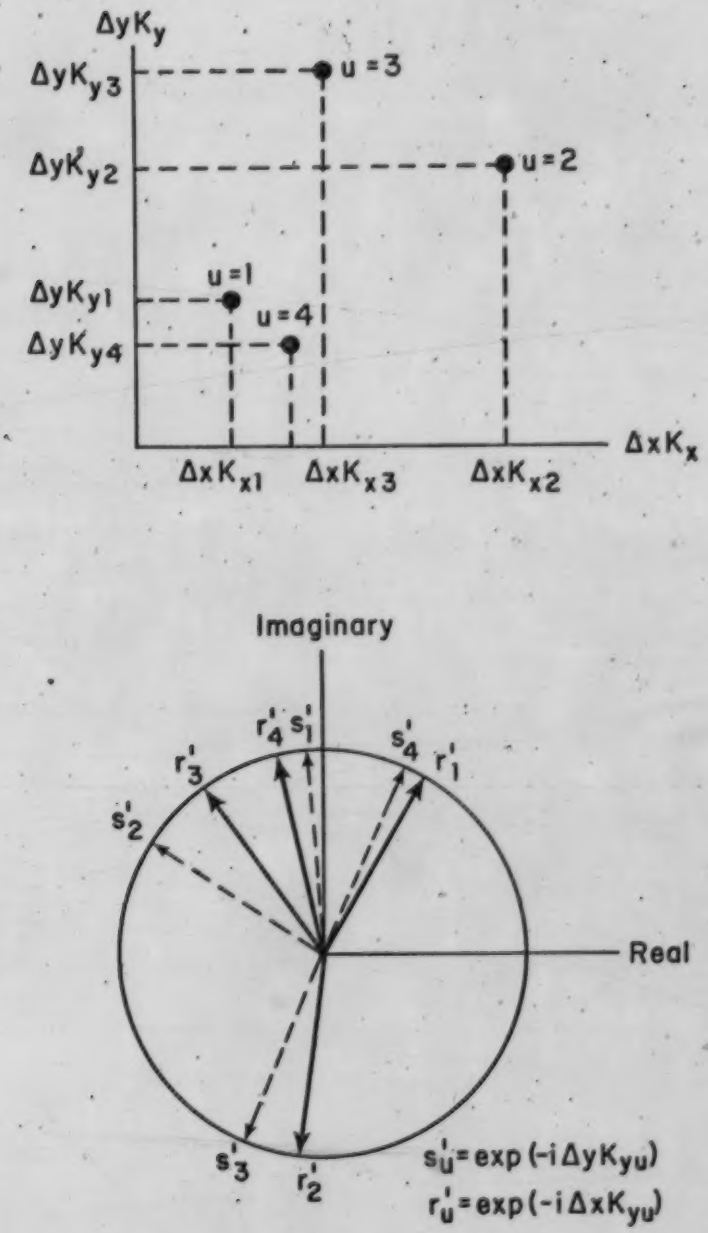


FIG. 4 ARBITRARILY CHOSEN PHASE ANGLES

rows identical, and consequently becomes singular (see 8.4.3 for ${}_mT(J_3)$). This requirement means

$$t_{(m-1)J_3+1} \neq t_{(m-1)J_3+2} \neq \ldots \neq t_{mJ_3} \quad \text{for} \quad m = 1,2,\ldots J_1J_2. \tag{9.2.1}$$

Sections 9.3 and 9.4 will show how these requirements can be met.

9.3 Creation of Column Redundancies by Selecting Phase Angles (The First J_3 Rows of P)

In this group, $1 \leq u \leq J_3$. From (8.3.1) and (8.3.2) we have $k_1 = 1$ and $k_2 = 1$.

9.3a Selection of ΔxK_x

(8.3.1) indicates that we require

$$r'_1 = r'_2 = \ldots = r'_{J3} . \tag{9.3.1}$$

To comply with this requirement we select

$$xK_{xu} = xK_{x1} + 2p_u\pi, \tag{9.3.2}$$

where p_u is an integer, and ΔxK_{x1} is a value to be defined later (e.g. see 9.5.1). At the moment ΔxK_{x1} may be considered as arbitrary. p_u should be different for different u in the first J_3 rows.

Substituting (9.3.2) into (8.2.2),

$$r'_u = \exp\{-i\Delta xK_{x1} - i2p_u\pi\} = \exp\{-i\Delta xK_{x1}\} \equiv r_{k1} \ . \qquad (9.3.3)$$

Let $u = 1,2,\ldots,J_3$ in (9.3.3), and $\underline{P}\{A\}$ indicate the principal value of a complex number A. If we set $k_1 = 1$, (9.3.3) becomes

$$r'_1 = r'_2 = \ldots = r'_{J3} = \underline{P}\{r'_1\} = r_1 \ . \qquad (9.3.4)$$

This is identical to (9.3.1) and to (8.3.1). (9.3.4) means that all the r'_u for $1 \le u \le J_3$ will have the same principal angle, therefore they are combined into one vector which we have named r_1 (See Fig. 5). ΔxK_{xu} then will not be randomly placed. Their values differ from one another by a multiple of 2π, as given in (9.3.2).

9.3b Selection of ΔyK_y

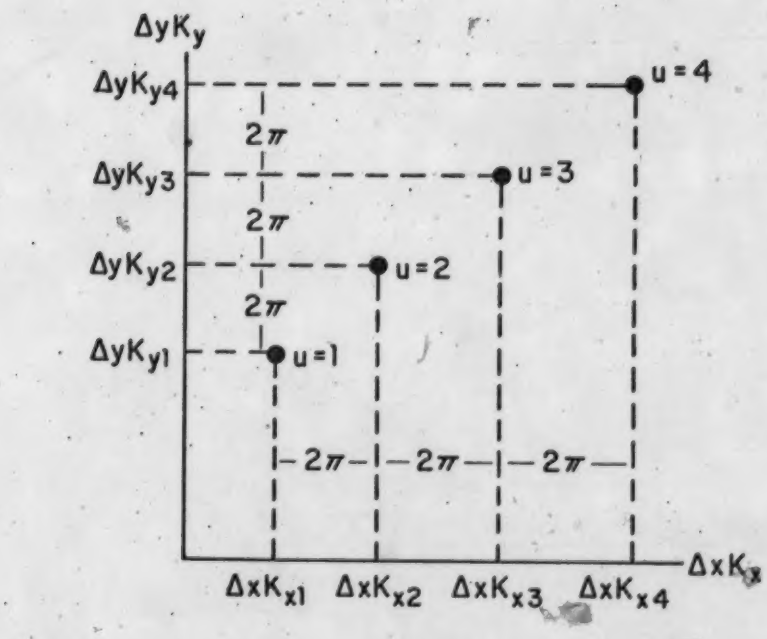
In a similar manner, we can choose

$$\Delta yK_{yu} = \Delta yK_{y1} + 2q_u\pi \ , \qquad (9.3.5)$$

where q_u is an integer, and ΔyK_{y1} is a constant to be defined later. Substituting the above into (8.2.2) yields

$$s'_1 = s'_2 = \ldots = s'_{J3} = \underline{P}\{s'_1\} \equiv s_{k2} = s_1 \ . \qquad (9.3.6)$$

This is identical to (8.3.2).



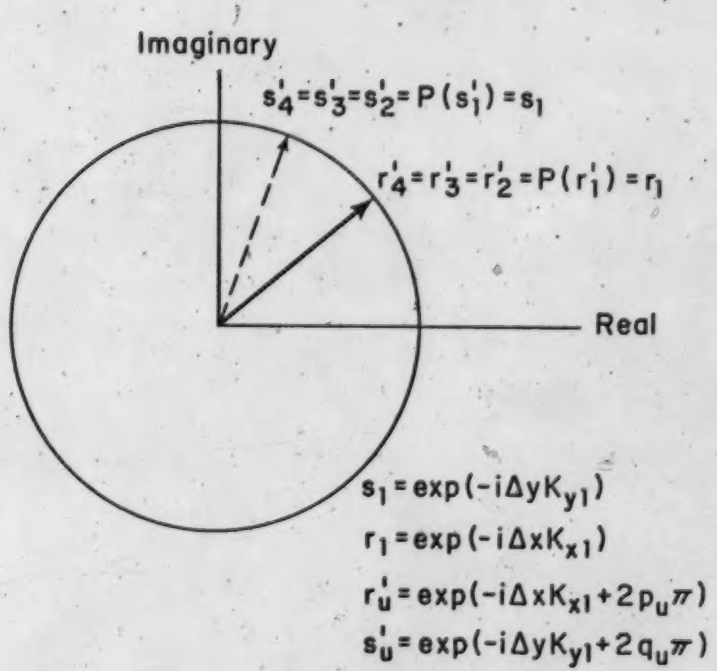


FIG. 5 A SPECIFICALLY CHOSEN K_u TO YIELD REDUNDANT FACTORS.

9.3c Restriction for Avoiding Singularity

(9.2.1) requires that all t_u must be different for $1 \leq u \leq J_3$. Since $t_u = \exp(-i\Delta z K_{zu}) = \exp(-i\Delta z \sqrt{K_o^2 - K_{xu}^2 - K_{yu}^2})$, (9.2.1) can be interpreted as:

(1) Among the J_3 sets of (K_{xu}, K_{yu}) values, any two sets having the same absolute values of K_x and K_y will yield a singular ${}_mT(J_3)$. That is to say, once the first vector K_1 is chosen,

$$|K_{xu}| = |K_{x1}| \quad \text{and} \quad |K_{yu}| = |K_{y1}| \qquad (9.3.7)$$

should not both be true.

(2) Interchange of absolute values of K_x and K_y will also result in a singular ${}_mT(J_3)$. That is to say,

$$|K_{xu}| = |K_{y1}| \quad \text{and} \quad |K_{yu}| = |K_{x1}| \; . \qquad (9.3.8)$$

should not both be true.

(9.3.7) and (9.3.8) can be illustrated in Fig.6. Once we have selected a point K_1, we may construct an octagon with the vertices symmetrical to K_1 with respect to the coordinate axes and also with respect to the two 45° inclined lines. Then the vertices are the

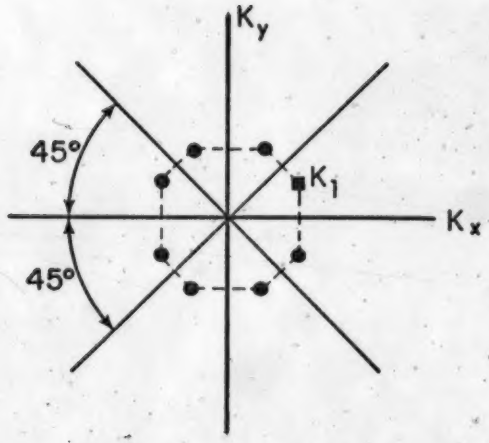


FIG. 6 THE FORBIDDEN OCTAGON VERTICES ON (K_x, K_y) PLANE

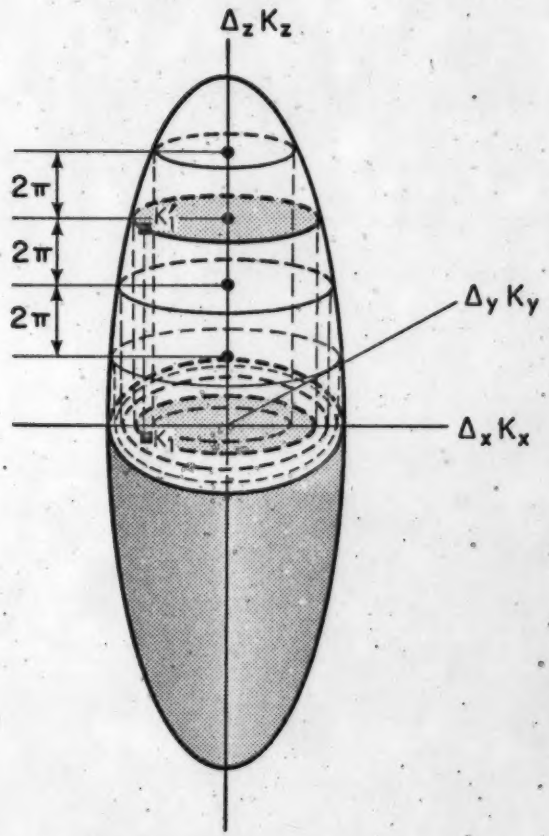


FIG. 7 THE FORBIDDEN ELLIPSES ON $(\Delta x K_x, \Delta y K_y)$ PLANE

forbidden points for K_u in the group of J_3 values of u. Only one vertex is permitted to be chosen.

(3) Nevertheless (9.2.1) would mean more forbidden points. Construct an axis ΔzK_z perpendicular to our $(\Delta xK_x, \Delta yK_y)$ plane. The three axes ΔxK_x, ΔyK_y and ΔzK_z form a phase domain. A sphere in the original K-domain (ref. Section 7.1) was described by

$$K_x^2 + K_y^2 + K_z^2 = k_o^2 \ . \qquad (9.3.9)$$

Since our axes have been rescaled, (9.3.9) becomes

$$\frac{(\Delta xK_x)^2}{\Delta x^2} + \frac{(\Delta yK_y)^2}{\Delta y^2} + \frac{(\Delta zK_z)^2}{\Delta z^2} = k_o^2 \qquad (9.3.10)$$

which is the equation of an ellipsoid in the phase domain (Fig. 7).

When ΔxK_{x1} and ΔyK_{y1} are selected the value ΔzK_{z1} is on the surface of the ellipsoid given by (9.3.10). Suppose another point K_u is to be selected. ΔzK_{zu} and ΔzK_{z1} should not differ by an integral multiple of 2π; otherwise we will have a singular ${}_mT(J_3)$. This gives the relation

$$\Delta zK_{zu} \neq \Delta zK_{z1} + 2d\pi \ , \quad d = \text{any integer.} \qquad (9.3.12)$$

As illustrated in Fig. 7, we have a family of forbidden ellipses on the $(\Delta xK_x, \Delta yK_y)$ plane. Only one point K_u, $1 \leq u \leq J_3$, is allowed to be selected on such a family of ellipses.

We have reason to suspect that (9.3.2) and (9.3.5) might contradict (9.3.7), (9.3.8) or (9.3.12). To clarify this, let us first consider the relation between (9.3.2), (9.3.5) and (9.3.12).

Theorem 9.3 Provided $\Delta x K_{x1}/\pi$ and $\Delta y K_{y1}/\pi$ are rational numbers, and $k_o^2 - K_{x1}^2 - K_{y1}^2$ not equal to a perfect square, the family of ellipses, containing the point $(\Delta x K_{x1}, \Delta y K_{y1})$ and represented by

$$\Delta z K_{zu} = \Delta z K_{z1} + 2d\pi \ , \quad d \text{ being an integer} \ , \tag{9.3.13}$$

will not contain the point $(\Delta x K_{xu}, \Delta y K_{yu})$ where

$$\left.\begin{aligned} \Delta x K_{xu} &= \Delta x K_{x1} + 2p\pi \\ \Delta y K_{yu} &= \Delta y K_{y1} + 2q\pi \end{aligned}\right\} , \quad p,q \text{ being integers} \ . \tag{9.3.14}$$

Proof: Substituting (9.3.12) and (9.3.14) into (9.3.10) gives

$$(K_{x1}+2p\pi/\Delta x)^2+(K_{y1}+2q\pi/\Delta y)^2+(K_{z1}+2d\pi/\Delta z)^2 \neq k_o^2. \tag{9.3.16}$$

Remember that (K_{x1}, K_{y1}, K_{z1}) satisfy (9.3.9). We expand (9.3.16) and divide the whole equation by $4\pi^2$.

$$(p/\Delta x^2)(p+\Delta x K_{x1}/\pi)+(q/\Delta y)^2(q+\Delta y K_{y1}/\pi)+(d/\Delta z^2)(d+\Delta z K_{z1}/\pi) \neq 0 \ . \tag{9.3.17}$$

We can see that (9.3.17) is generally true, because $\Delta x K_{x1}/\pi$, $\Delta y K_{y1}/\pi$ are assumed rational; p, q, d are

integers; $p/\Delta x^2$, $q/\Delta y^2$, $d/\Delta z^2$ are rational numbers. The only irrational term is $\Delta z K_{zl}/\pi = \Delta z\sqrt{k_o^2-K_{xl}^2-K_{yl}^2}/\pi$. Hence all the terms in (9.3.9) will not be summed up to zero. This proves that (9.3.14) does not contradict (9.3.13). In other words, (9.3.2) and (9.3.5) present no contradiction with (9.3.12) or (9.3.11).

Now we consider the relation between (9.3.2), (9.3.5) and (9.3.7). These three equations indicate that we may have trouble when

$$\Delta x K_{xu} = \Delta x K_{xl} + 2p\pi = \pm \Delta x K_{xl} \quad \text{and}$$
$$\Delta y K_{yu} = \Delta y K_{yl} + 2q\pi = \pm \Delta y K_{yl} \quad \text{are both true.} \tag{9.3.18}$$

Solving (9.3.18) shows that we may have trouble when

$$\text{either } p = 0 \quad \text{or} \quad \Delta x K_{xl} = -\pi p, \quad \text{and}$$
$$\text{either } q = 0 \quad \text{or} \quad \Delta y K_{yl} = -\pi q \quad \text{are both true.} \tag{9.3.19}$$

The first equation in (9.3.19) gives K_u symmetrical to K_l with respect to the $\Delta x K_x$ axis. To avoid this trouble, we can either set $\Delta x K_{xl}$ not equal to integral multiple of π; or simply select K_l and K_u on the right (or on the left) half of the $(\Delta x K_x, \Delta y K_y)$ plane (Fig.8). The second equation in (9.3.19) shows K_u symmetrical to K_l with respect to the $\Delta y K_y$ axis, and this symmetry can be avoided if $\Delta y K_{yl}$ is not equal to integral multiple of

π. We can also select K_l and K_u in the upper (or the lower) half of the coordinate plane.

Finally, let us consider the relation between (9.3.2), (9.3.5) and (9.3.8). These equations indicate that we may have trouble if

$$\Delta xK_{xu} = \Delta xK_{xl} + 2p\pi = \pm \Delta yK_{yl} \quad \text{and}$$
$$\Delta yK_{yu} = \Delta yK_{yl} + 2q\pi = \pm \Delta xK_{xl} \quad \text{are both true.} \tag{9.3.20}$$

(9.3.20) is equivalent to say: we should avoid

$$\Delta xK_{xl} \pm \Delta yK_{yl} = -2p\pi \text{ or } 2q\pi \ .$$

In other words, if the difference between (or sum of) ΔxK_{xl} and ΔyK_{yl} is integral multiple of 2π, K_u will be symmetrical to K_l with respect to one of the 45° inclined lines (see Fig. 8). To avoid this symmetry, we should either avoid the difference or sum of the coordinates of K_l being a 2π multiple, or select all the K_u points in a quadrant of the (ΔxK_x, ΔyK_y) plane formed by two 45° inclined lines.

To summarize the above, we should avoid taking ΔxK_{xl} and ΔyK_{yl} equal to integral multiples of π. If the worst comes to the worst, this being unavoidable, we can select the K_u points in the first octant of the (ΔxK_x, ΔyK_y) plane (Fig. 8).

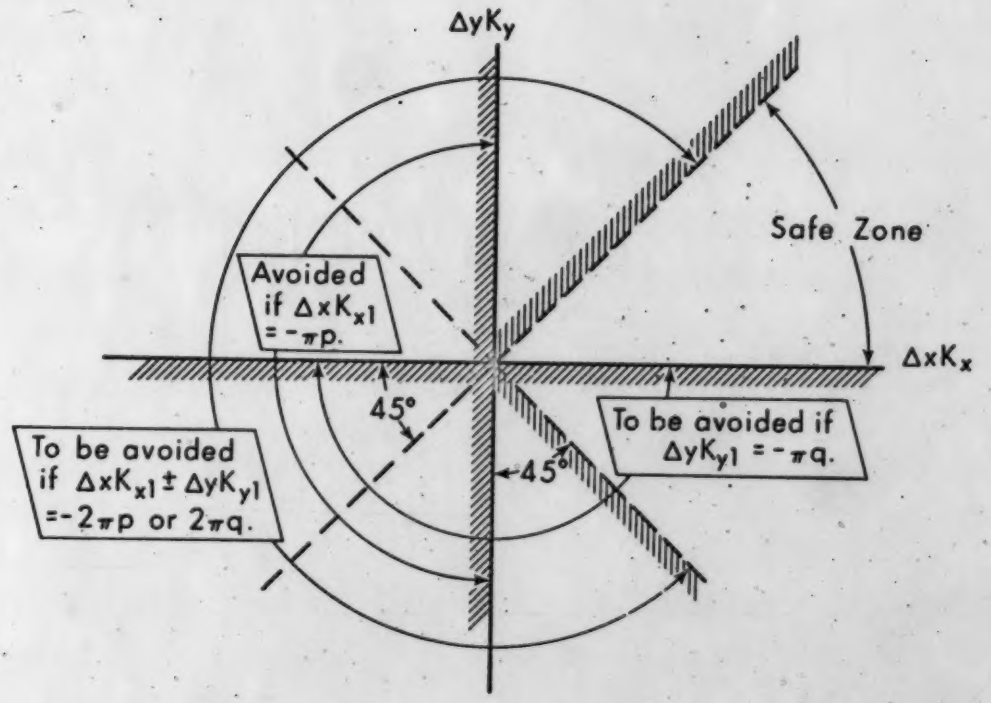


FIG. 8

ZONES TO BE AVOIDED IF THE COORDINATES K_{x1}, K_{y1} OF THE PREVIOUS POINT ARE CHOSEN SUCH THAT EITHER $\Delta x K_{x1}$ OR $\Delta y K_{y1}$ IS EQUAL TO A MULTIPLE OF π

9.4 Creation of Column Redundancies by Selecting Phase Angles (the Remaining Rows of P)

For the second group of J_3 values of K, namely the K_u for $J_3+1 \leq u \leq 2J_3$, selection will proceed in a similar manner. The only two differences are: s'_{J3+1} is no longer arbitrary and should be equal to the s_1 value selected previously in Section 9.3; and r'_{J3+1} is arbitrary but should not be equal to r_1 in order to avoid a singular $R(J_1)$.

$$s'_{J3+1} = s'_{J3+2} = \ldots = s'_{2J3} = \underline{P}\{s'_1\} = s_1$$
$$r'_{J3+1} = r'_{J3+2} = \ldots = r'_{2J3} = \underline{P}\{r'_{J3+1}\} = r_2 \neq r_1. \qquad (9.4.1)$$

For the third group, the same way of selection of K_u applies, and we have another group of J_3 s'_u all equal to the s_1 determined in Section 9.3. We also have another group of J_3 r'_u all equal to r_3, which differs from r_1 and r_2.

In this manner, K_u will be selected for J_1 groups of J_3 rows in P. The result is that we have only one value s_1 for all these rows, and J_1 different values of r_{k1}, $k_1 = 1,2,\ldots,J_1$.

Then we shall select K_u for the next J_1 groups of J_3 rows in P. We shall have an s_2, which is different from s_1, for all these rows. However, the r'_u will not be randomly chosen. r'_u will be in J_1 groups of J_3

values each, and they should be equal to $r_1, r_2, \ldots,$ r_{J1} respectively.

We shall proceed until we assign all K_u for all the rows. When this process is completed, we shall have J_2 different values of s_{k2} and J_1 different values of r_{k1}. We have so far no restrictions for these values. But we understand that once we have chosen the J_1 ΔxK_x values and J_2 ΔyK_y values for r and s, all the K_u will be determined by our method just described.

9.5 Stabilizing the Vandermonde Matrices

From Sections 9.3a through 9.3c we have set a definite pattern for selecting phase angles for matrix P. The pattern is

$$\Delta xK_{xu} = \Delta xK_{x1} + 2p_u\pi, \text{ and}$$
$$\Delta yK_{yu} = \Delta yK_{y1} + 2q_u\pi. \qquad (9.5.1)$$

We know that the selection of ΔxK_{xu} and ΔyK_{yu} will eventually result in a series of matrices ${}_mT(J_3)$, $m = 1,2,\ldots,J_1J_2$, which (see 8.4.3) are Vandermonde matrices belonging to Sequence C. To avoid a singular ${}_mT(J_3)$ being created, ΔxK_{xu} and ΔyK_{yu} should in general not equal a multiple of π (see Section 9.3c).

We have so far not been concerned with the actual values of ΔxK_{x1} and ΔyK_{y1}, hence the actual values of ΔxK_{xu} and ΔyK_{yu}. These actual values will be discussed in Section 9.6 and Chapter XI.

We have not yet mentioned the choice of values p_u and q_u, although on some occasions we said that p_u and q_u are both integers. In this section we shall describe how to find the optimal p_u and q_u to guide our choice.

To start the description, let us refer to Section 4.2, in which it was indicated that Vandermonde matrices are known to be ill-conditioned and attention should be paid to instability. Our aim is to select p_u and q_u such that ${}_mT(J_3)$ can be stabilized to the best possible extent.

The inversion of Sequence C is given in (4.2.5). The matrix $[\underline{D}(n)D^{-1}(n)]$ contains factors (x_1-x_2), (x_2-x_3), (x_1-x_3), (x_2-x_4) ... etc. If x_1 is too closed to x_2 in value, then $[\underline{D}(n)D^{-1}(n)]$ will have most of its diagonal elements containing very small denominators, consequently the inversion of ${}_mT(J_3)$ becomes unstable. Instability will also occur when any one of the $x_1, x_2, \ldots, x_{J3}$ is equal to any of the others. To avoid this, we have to maximize the differences between any pair of them.

In our case, the elements of ${}_mT(J_3)$ are denoted by $t_1, t_2, \ldots, t_{J3}$. We select any two consecutive t's

and call them t_k and t_j. Since

$$t_u = \exp(-i\,\Delta z\, K_{zu}) , \tag{9.5.2}$$

t_u is a complex number. To maximize the difference between all the t's would mean to separate the t's with equal angles within 2π (see Figs. 9 and 10):

$$\Delta z\, K_{zj} - \Delta z\, K_{zk} = 2\pi(j-k)/J_3 . \tag{9.5.3}$$

Substituting K_{xu}, K_{yu} for K_{zu}, dividing (9.5.3) by Δz, and using (9.5.1),

$$\sqrt{k_o^2-(K_{x1}+2p_j\pi/\Delta x)^2 - (K_{y1}+2q_j\pi/\Delta y)^2} -$$

$$\sqrt{k_o^2-(K_{x1}+2p_k\pi/\Delta x)^2 - (K_{y1}+2q_k\pi/\Delta y)^2} = 2\pi(j-k)/(\Delta z J_3). \tag{9.5.4}$$

Moving the first square-root term to the right of (9.5.4), squaring both sides and cancelling k_o^2,

$$-(K_{x1}+2p_k\pi/\Delta x)^2 - (K_{y1}+2q_k\pi/\Delta y)^2 =$$

$$(2\pi[j-k]/\Delta z J_3)^2 - (K_{x1}+2p_j\pi/\Delta x)^2$$

$$- (K_{y1}+2q_j\pi/\Delta y)^2 - 4\pi K_{zj}(j-k)/(\Delta z J_3). \tag{9.5.5}$$

Cancelling K_{x1}^2 and K_{y1}^2 terms,

$$4\pi(p_j-p_k)K_{x1}/\Delta x + 4\pi(q_j-q_k)K_{y1}/\Delta y + 4\pi^2\{(p_j^2-p_k^2)/\Delta x^2$$

$$+(q_j^2-q_k^2)/\Delta y^2\} - 4\pi^2(j-k)^2/(\Delta z J_3)^2 + 4\pi K_{zj}(j-k)/(\Delta z J_3) = 0. \tag{9.5.6}$$

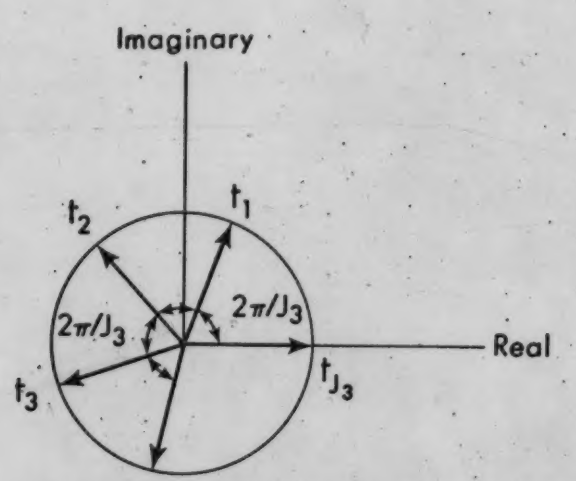


FIG. 9 OPTIMIZED VECTORS $t_1, t_2, \cdots t_{J_3}$ SUBTEND EQUAL ANGLES AND HAVE UNITY MODULI

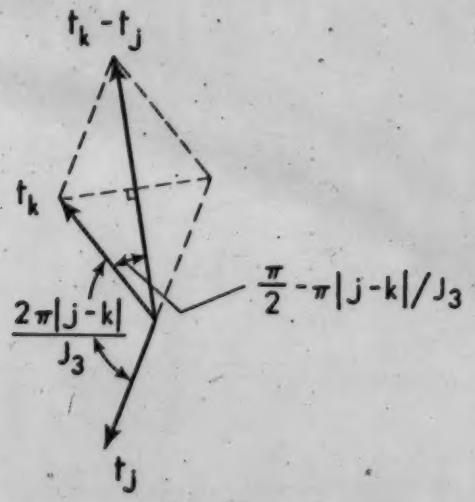


FIG. 10 ILLUSTRATION OF $t_k - t_j$

For simplicity, let us assume that we have selected K_1 and we wish to see how we are going to select $K_2, K_3 \ldots$ to maximize the stability. In this case, $p_k = 0$, $q_k = 0$. We can write p_j as p and q_j as q. We also assume that $\Delta x = \Delta y = \Delta z$. (9.5.6) becomes

$$p\Delta xK_{x1} + q\Delta yK_{y1} + \pi p^2 + \pi q^2 + \pi/J_3^2 - \Delta zK_{z1}/J_3 = 0 \ . \qquad (9.5.7)$$

Solving for q,

$$q_o = \frac{-\Delta yK_{y1} \pm \sqrt{(\Delta yK_{y1})^2 - 4\pi(p_o\Delta xK_{x1} + \pi p_o^2 + \pi/J_3^2 - \Delta zK_{z1}/J_3)}}{2\pi} \qquad (9.5.8)$$

We write p as p_o and q as q_o because the value obtained is an optimal value, not the actual selected value of p and q.

As a numerical example, we have the following experimental data:

$$\begin{aligned} &\lambda = 6200 \text{ Å} \\ &\Delta x = \Delta y = \Delta z = 10 \text{ micron} = 10^{-5} \text{ meter} \\ &J_3 = 4 \\ &\Delta xK_{x1} = 0.8 \text{ radians} \\ &\Delta yK_{y1} = 0.8 \text{ radians} \end{aligned} \qquad (9.5.9)$$

Then

$$k_o = 2\pi/\lambda \doteq 10^7 \text{ meter}^{-1}$$
$$k_{z1} = 10^7 \text{ meter}^{-1} \qquad (9.5.10)$$
$$\Delta z K_{z1} = 100 \text{ radians.}$$

Since $\Delta z K_{z1} >> \Delta x K_{x1}$ and $\Delta z K_{z1} >> \Delta y K_{y1}$, we can ignore the terms involving $\Delta x K_{x1}$ and $\Delta y K_{y1}$ in (9.5.8), leaving

$$q_o \approx \pm\sqrt{\Delta z K_{z1}/J_3\pi - p_o^2 - 1/J_3^2} \qquad (9.5.11)$$

$$\approx \sqrt{7.89 - p_o^2}\ , \quad \text{or}$$

$$q_o^2 + p_o^2 \approx 7.89\ . \qquad (9.5.12)$$

p and q must be integers; an ideal solution for the above would be $p_o = 2$ and $q_o = 2$. This yields an optimal K_u distribution on the $(\Delta x K_x, \Delta y K_y)$ plane. From (9.5.1) and (9.5.9):

$$\begin{aligned}
&\Delta x K_{x1} = \text{arbitrary}\ , && \Delta y K_{y1} = \text{arbitrary} \\
&\Delta x K_{x2} = \Delta x K_{x1} + 4\pi\ , && \Delta y K_{y2} = \Delta y K_{y1} + 4\pi \\
&\Delta x K_{x3} = \Delta x K_{x2} + 4\pi\ , && \Delta y K_{y3} = \Delta y K_{y2} + 4\pi \\
&\quad\cdots\cdots && \quad\cdots\cdots \\
&\Delta x K_{xn} = \Delta x K_{x1} + (n-1)4\pi, && \Delta y K_{yn} = \Delta y K_{y1} + (n-1)4\pi\ .
\end{aligned} \qquad (9.5.13)$$

When choosing p and q in this way, we should not forget that the values of $\Delta x K_x$ and $\Delta y K_y$ must represent a point on the ellipsoid. Thus,

$$(\Delta x K_{xu}/\Delta x)^2 + (\Delta y K_{yu}/\Delta y)^2 \le k_o^2 \quad \text{for all u}\ . \qquad (9.5.14)$$

If this requirement is not met, p and q will be less than p_o and q_o.

9.6 Factor Matrices Belonging to the E Sequence

So far the values $r_1 \ldots . r_{J1}$, $s_1 \ldots . . s_{J2}$, corresponding to $\Delta x K_{x((k_1-1)J_3+1)}$ and to $\Delta y K_{y((k_2-1)J_1J_3+1)}$ for $k_1 = 1,2,\ldots J_1$, $k_2 = 1,2,\ldots J_2$, remain free. We can take advantage of these values to further expedite our computation. We know that $r_1 \ldots . r_{J1}$ are the first columns of matrix $R(J_1)$ and $s_1 \ldots . s_{J2}$ are those of $S(J_2)$ in (8.4.9). In our fast algorithm (8.4.10), $R(J_1)$ has to be inverted and $S(J_2)$ has to be inverted and transposed. Perhaps a clever choice of these will make the inversions easy.

One way to achieve this is to select (for another way, see Chapter XI)

$$\Delta x K_{x((k_1-1)J_3+1)} = \pi(2k_1-1)/J_1 \ . \qquad (9.6.1)$$

The result of the selection is

k_1	1	2	....	J_1
$\Delta x K_x$	π/J_1	$3\pi/J_1$	....	$\pi(2J_1-1)/J_1$
r_{k1}	$\exp(-i\pi/J_1)$	$\exp(-i3\pi/J_1)$	....	$\exp(-i(2J_1-1)\pi/J_1)$

This selection has avoided $\Delta xK_x = -p\pi$, thus avoided the ΔxK_x-axis symmetry (ref. 9.3.19). Comparing the above with (4.3.1), we can see that r_{k1} are identical to the first column of $E(J_1)$ in the E sequence. This is followed immediately by the easy inversion

$$R^{-1}(J_1) = \hat{R}(J_1) \ . \tag{9.6.2}$$

Similarly we shall select

$$\Delta yK_{y((k_2-1)J_1J_3+1)} = \pi(2k_2-1)/J_2 \ . \tag{9.6.3}$$

This selection will avoid the ΔyK_y-axis symmetry and

$$S^{-t}(J_2) = \tilde{S}(J_2) \ . \tag{9.6.4}$$

Therefore the solution to Pf = F in (8.4.10) becomes

$$\text{Rect}_{J2}\ \text{Rect}_{J3}\ f = \hat{R}(J_1)\ \text{Rect}_{J2}\ F''\ \tilde{S}(J_2)/J_1J_2. \tag{9.6.5}$$

Let us split J_3 into 2 factors J_4 and J_5:

$$J_3 = J_4J_5, \quad J_4, J_5 \text{ being positive integers} \ . \tag{9.6.6}$$

Then let

$$p_u=(k_3-1)p_o, \quad k_3=\text{Int}\{(u-1)\text{mod}(J_3)/J_4\}+1=1,2,\ldots,J_5 \ . \tag{9.6.7}$$

$$q_u=(k_4-1.75)q_o, \quad k_4=(u-1)\text{mod}(J_4)+1=1,2,\ldots,J_4. \tag{9.6.8}$$

From (9.3.2) and (9.3.5),

$$\Delta xK_{xu} = \pi(2k_1-1)/J_1 + 2(k_3-1)p_o\pi$$

$$\Delta yK_{yu} = \pi(2k_2-1)/J_2 + 2(k_4-1.75)q_o\pi \qquad (9.6.9)$$

$$u = k_3 + (k_4-1)J_5 + (k_1-1)J_5J_4 + (k_2-1)J_5J_4J_1 \; . \qquad (9.6.10)$$

A numerical example will be given in the next chapter. The following is a summary of selection of parameters:

J_1, J_2, J_3 are determined by the physical model in Fig. 3.

J_4, J_5 are determined by (9.6.6).

$k_1, k_2, k_3, k_4, p_u, q_u$ are determined by (8.3.1), (8.3.2), (9.6.7), (9.6.8), where p_o and q_o are obtained from (9.5.8), and $u = 1,2,\ldots J_1J_2J_3$.

j_1, j_2, j_3 can be calculated from (8.2.2) for $v = 1,2,\ldots J_1J_2J_3$.

ΔxK_{xu} and ΔyK_{yu} are obtained from (9.6.9), ΔzK_{zu} from (9.3.10).

From the above we shall be able to determine r,s,t, R,S,T,L, etc..

The method of parameter selection helps to achieve the following:

(1) The matrix P created will comply with the format given by (8.3.4).

(2) Singularity in R, S or $_mT$ matrices has been avoided.

(3) Stability in the $_mT$ matrices is optimized.

(4) Matrices R and S belong to the E sequence, and are easy for inversion.

(5) The matrices $_mT$, hence matrix P, can be quickly inverted.

(6) We shall see in Chapter XII, that the best condition number for P is obtained through selecting phase angles in the ways described above.

X. A COMPUTED EXAMPLE

10.1 A Simplified Physical System

Let $J_1 = J_2 = J_3 = 4$. The scattering object is represented by $4 \times 4 \times 4 = 64$ grid points, to each of which is attached a scattered field source density f_v. For the sake of easy comprehension, only 8 grid points are assigned a non-zero source field density $f = 2+i$ where $i^2 = -1$, and at the other points the source densities are assumed nil. Fig. 11 shows the actual assumed positions of non-zero source fields as follows:

$$f_v = \begin{cases} 2+i & \text{for } v = 34,35,46,47,50,51,62,63. \\ 0 & \text{for other values of } v\ . \end{cases} \qquad (10.1.1)$$

We know we have continuous far-field data available. Let us see how we are going to choose a K_u which leads us to select the data F_u at proper points and determine the matrix P for a quick solution of $Pf = F$.

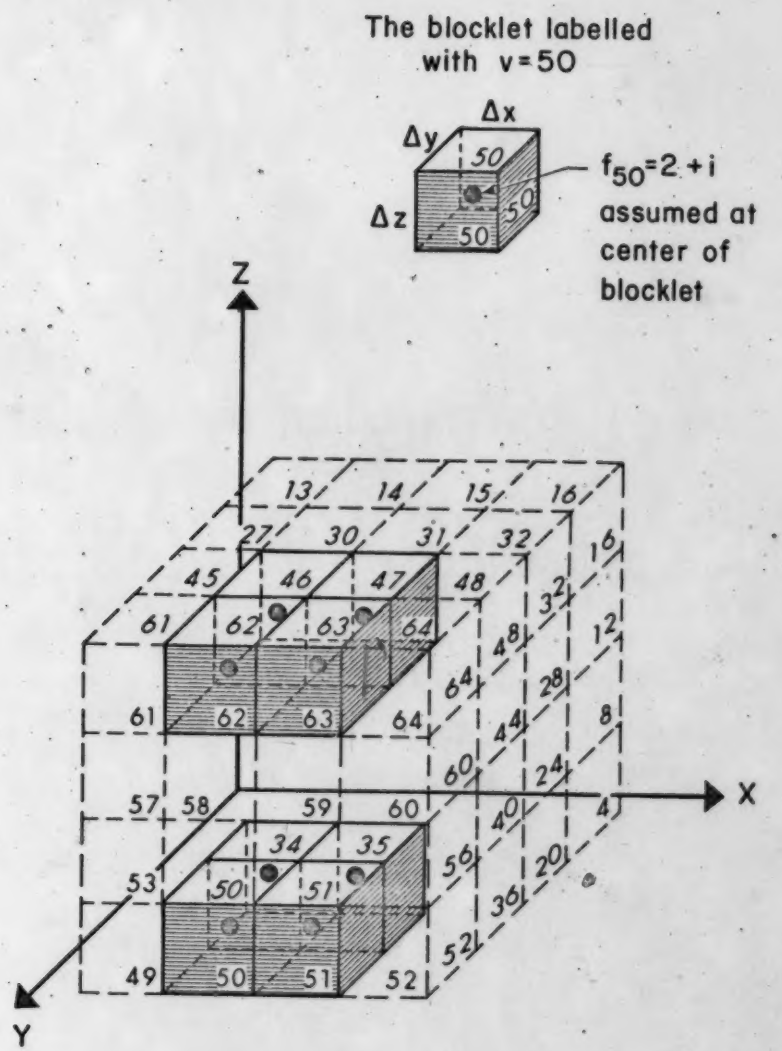
10.2 Selection of Phase Angles

Let $J_3 = J_4 J_5 = 2 \times 2$. From (9.6.10) we have

$$u = k_3 + 2(k_4-1) + 4(k_1-1) + (k_2-1)(2\times 2\times 4)$$

$$= k_3 + 2k_4 + 4k_1 + 16k_2 - 22. \qquad (10.2.1)$$

The k's will be computed from u according to (8.3.1),



$$\begin{cases} f_v = 2+i & \text{for } v = 34, 35, 46, 47, 50, 51, 62, 63; \\ f_v = 0 & \text{for other values of } v. \end{cases}$$

FIG. 11 THE SCATTERER IS REPRESENTED BY EIGHT NONZERO SOURCE DENSITY SAMPLES

(8.3.2), (9.5.7), (9.5.8):

u	k_1	k_2	k_3	k_4
1	1	1	1	1
2			2	
3			1	2
4			2	
5	2		1	1
6			2	
7			1	2
8			2	
9	3	2	1	1
10			2	
11			1	2

u	k_1	k_2	k_3	k_4
	2	3		2
56			2	
57	3	4	1	1
58			2	
59			1	2
60			2	
61	4		1	1
62			2	
63			1	2
64			2	

Assuming $\Delta x = \Delta y = \Delta z$ and $\Delta z k_o = 62.8$, from (9.6.1), (9.3.10) and (9.5.11) we have

$$\Delta x K_{x1} = 0.25\pi, \quad \Delta y K_{y1} = 0.25\pi, \quad \text{then } \Delta z K_{z1} \approx 62,$$

$$q_o \approx \pm\sqrt{62/4\pi - p_o^2 - 1/16} \approx \sqrt{4.9 - p_o^2}\ .$$

A suitable pair of solution will be $q_o = 1$ and $p_o = 2$. From (9.6.9),

$$\begin{cases} \Delta x K_{xu} = -4.25\pi + 4\pi k_3 + 0.5\pi k_1 \\ \Delta y K_{yu} = -2.25\pi + 2\pi k_4 + 0.5\pi k_2. \end{cases} \tag{10.2.2}$$

By assigning the k values as given in the Table above we have 64 values of K_u, the projections of which onto the $(\Delta xK_x, \Delta yK_y)$ plane form a selection pattern as shown in Fig. 12. The actual values of K_u are calculated by computer and are listed in Appendix II. ΔzK_z is calculated according to

$$\Delta zK_{zu} = \Delta z \sqrt{k_o^2 - K_{xu}^2 - K_{yu}^2} \quad . \tag{10.2.3}$$

Computation of r_{k1}, s_{k2} can be done using (9.3.4) and (9.3.6). Computation of t_u will be in accordance with (8.2.2). These values form the matrices R(4), S(4) and ${}_mT(4)$. We store these matrices instead of storing P to save space.

In this example, we have

$$r_{k1} = \exp[-i\Delta xK_{xu}] = \exp[-i\pi(-4.25+4k_3+0.5k_1)] \tag{10.2.4}$$

$$= \exp[-i\pi(0.5k_1-0.25)], \quad k_1 = 1,2,3,4.$$

$$s_{k2} = \exp[-i\Delta yK_{yu}] = \exp[-i\pi(-2.25+2k_4+0.5k_2)] \tag{10.2.5}$$

$$= \exp[-i\pi(0.5k_2-0.25)], \quad k_2 = 1,2,3,4.$$

Therefore $r_{k1} = s_{k2}$ or R(4) = S(4). The generated data in Appendix III give

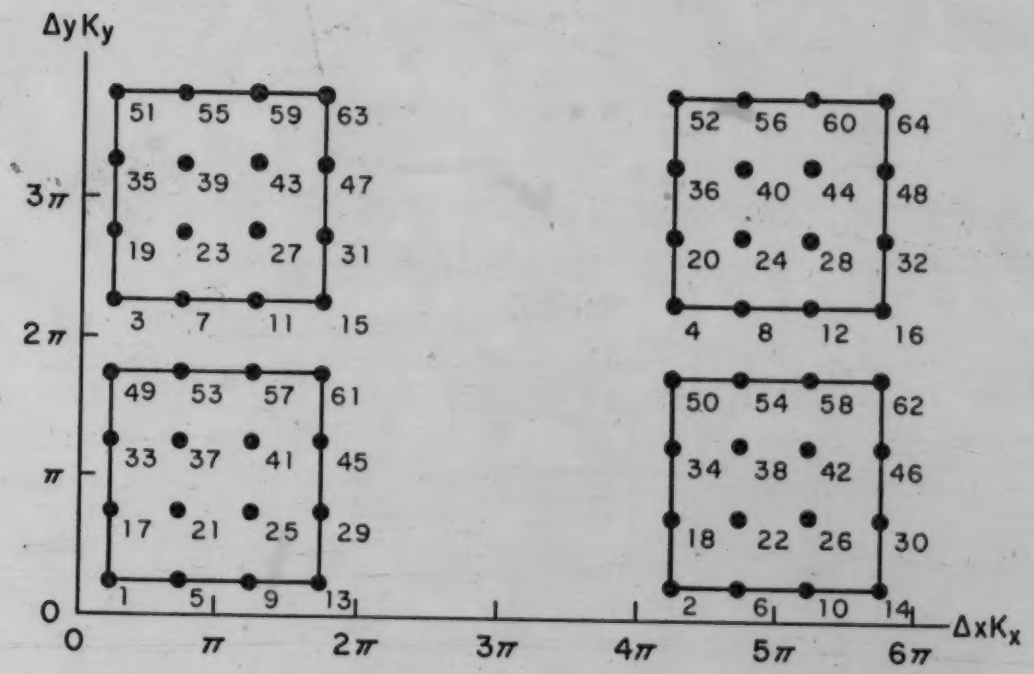


FIG. 12

A SELECTION PATTERN FOR $K_1, K_2, \ldots, K_{64}$
AS PROJECTED ONTO THE $(\Delta x K_x, \Delta y K_y)$ PLANE

(The numbers attached to each point are the values of u)

$$R(4) = \begin{bmatrix} \sqrt{2}(1-i)/2 & -i & \sqrt{2}(-1-i)/2 & -1 \\ \sqrt{2}(-1-i)/2 & i & \sqrt{2}(1-i)/2 & -1 \\ \sqrt{2}(-1+i)/2 & -i & \sqrt{2}(1+i)/2 & -1 \\ \sqrt{2}(1+i)/2 & i & \sqrt{2}(-1+i)/2 & -1 \end{bmatrix} = E(4) \quad (10.2.6)$$

We can be sure that the inverse of R(4) or S(4) is $\hat{R}(4)/4$ and the ordinary inversion algorithms are not needed.

The ${}_mT(4)$ matrices, for m = 1,2,...,16 are also listed in Appendix III. Every ${}_mT(4)$ is a C(4) matrix. Its second column equals the square of its first column, and its third column equals the cube of its first column, etc. Hence our fast inversion algorithm for sequence C will apply. Furthermore, it can be verified, from the numerical data, that t_1, t_2, t_3 and t_4 in every ${}_mT(4)$ subtend big angles between them, so that ${}_mT(4)$ becomes well-conditioned as a result of adopting the optimized values for p_u and q_u.

F will be simulated, using the R, S, ${}_mT$ and the f values in (10.1.1). The simulated F will be our data for the system Pf = F.

Then we solve the system Pf = F by our fast algorithm to obtain f, and compare the obtained f with the f in (10.1.1).

10.3 Computer Programs

Our computing program is written in three parts:

(A) A program to generate K·X components, R(4), S(4), ${}_mT(4)$ and F_u, and to store the data in file DATA.

(B) A program to solve Pf = F by fast algorithm. This program is compiled and the object program is stored in file COMPUT.

(C) An execution program to run COMPUT using data in DATA.

The program in full is attached in Appendix III. Computer outputs are also enclosed. It is interesting to note that to solve Pf = F using a set of given data requires only program (C), which needs 2.2 seconds CPU time for $J_1 = J_2 = J_3 = 4$. The maximum error shown in the computed f_v does not exceed 0.0006%, which should be acceptable to both scientific research and the optical industry.

XI. VARIATIONS OF THE METHOD OF SOLUTION

11.1 Factor Matrices Belonging to the Fourier Transform Sequence

The algorithm described in previous chapters has been based on formula (8.4.10), namely,

$$\text{Rect}_{J2}\ \text{Rect}_{J3}\ f = R^{-1}(J_1)(\text{Rect}_{J2}F'')\ S^{-t}(J_2). \qquad (11.1.1)$$

When R and S belong to the E sequence, we have (9.5.5):

$$\text{Rect}_{J2}\ \text{Rect}_{J3}\ f = \hat{R}(J_1)(\text{Rect}_{J2}F'')\tilde{S}(J_2)/J_1J_2. \qquad (11.1.2)$$

As an alternative method, we can replace (9.6.1) and (9.6.3) by

$$\Delta x K_x((k_1-1)J_3+1) = 2\pi k_1 i/J_1\ ;\ \Delta y K_y((k_2-1)J_1J_3+1 = 2\pi k_2 i/J_2. \qquad (11.1.3)$$

Then from (4.4.3)

$$\begin{aligned} R(J_1) &= \underline{F}^*(J_1) \\ S(J_2) &= \underline{F}^*(J_2)\ . \end{aligned} \qquad (11.1.4)$$

The $\underline{F}$ matrices are members of the sequence described in (4.4.1). From (4.4.3) we have

$$R^{-1}(J_1) = \underline{F}(J_1)/J_1 \qquad (11.1.5)$$

$$S^{-t}(J_2) = \underline{F}(J_2)/J_2\ . \qquad (11.1.6)$$

We conclude that

$$\text{Rect}_{J2}\ \text{Rect}_{J3}\ f = \underline{F}(J_1)\,(\text{Rect}_{J2}F'')\,\underline{F}(J_2)/J_1J_2. \qquad (11.1.7)$$

When $J_1 = J_2 = J_3 = 4$, the selection pattern of K_u is shown in Fig. 13, which is similar to Fig. 12 except that all points are shifted by a distance $(\pi/4, \pi/4)$.

Since (11.1.7) is as simple as (11.1.2), the number of multiplications required will be the same, i.e., $J_1J_2(4.5J_3^2+J_1+J_2+J_3)$. As far as storage space is concerned, we can store a single w (see 4.4.1) instead of storing the whole Fourier transform matrix. Hence the storage of P can be simplified to storing J_1J_2 matrices of ${}_mT$, and each ${}_mT$ needs only to store its first column, as the other columns of a Vandermonde matrix are easily generated from the first column elements.

Andrews (1970) called the Kronecker matrix L, defined by

$$L = S \otimes R = \underline{F}^* \otimes \underline{F}^* \qquad (11.1.8)$$

a generalized Walsh transform matrix, which is orthogonal. An alternative solution of $Lf = F''$ comes directly from the orthogonality and from (5.2.4):

$$\text{Rect}_{J3}\ f = [\underline{F} \otimes \underline{F}]\ F''/J_1J_2\ . \qquad (11.1.9)$$

This gives us an algorithm no faster than (11.1.7). We have already explained this in Section 6.5.

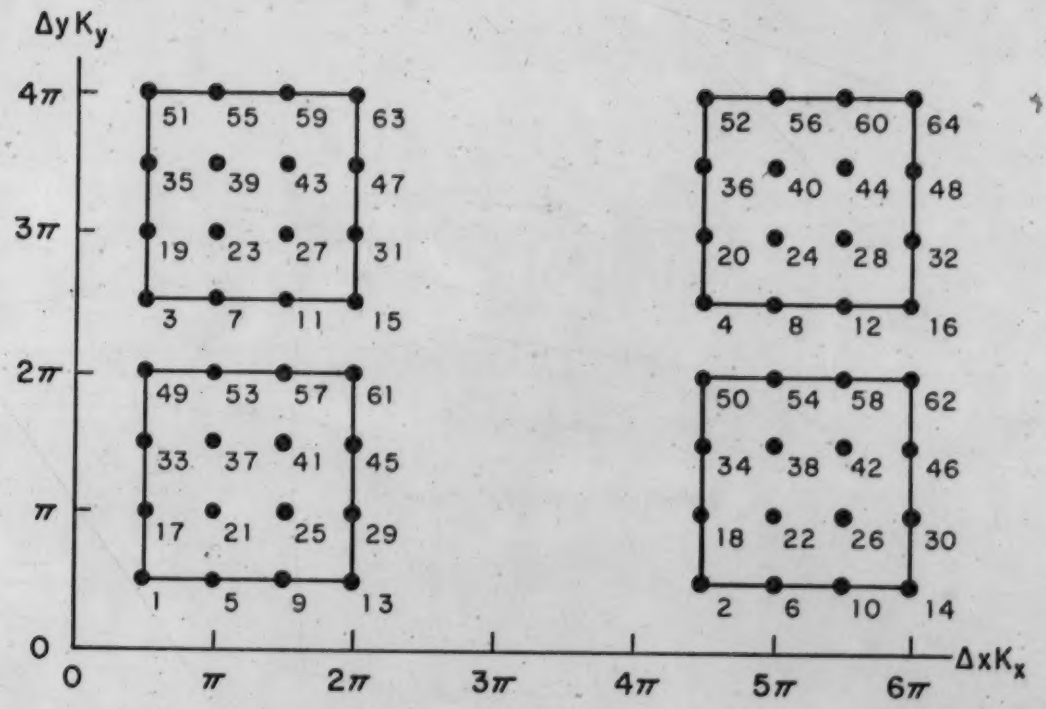


FIG. 13

AN ALTERNATIVE SELECTION PATTERN FOR $K_1 \ldots K_{64}$, USING THE FOURIER TRANSFORM SEQUENCE.

(The numbers attached to each point are the values of u)

PART THREE

ERROR ANALYSIS

XII. SEQUENCE CONDITION NUMBERS AND ERROR ANALYSIS

12.1 Turing's N-Condition Number

The relative error in solving a system of equations is related to the errors in the system's coefficients and in the constant vectors. With ΔX, ΔY, ΔM denoting the errors in systems $MX = \overline{Y}$, the relation given, for instance by Isaacson and Keller (1966), is

$$\frac{||\Delta X||}{||X||} \leq \frac{||M^{-1}||\ ||M||}{1 - ||\Delta M||\ ||M^{-1}||} \left(\frac{||\Delta Y||}{||Y||} + \frac{||\Delta M||}{||M||} \right) . \qquad (12.1.1)$$

The validity of the above is subject to $||\Delta M||\ ||M^{-1}|| < 1$. In usual practice, ΔM is rather small so that we have

$$||\Delta M||\ ||M^{-1}|| << 1 . \qquad (12.1.2)$$

Then (12.1.1) may be replaced by the following:

$$\frac{||\Delta X||}{||X||} \leq ||M^{-1}||\ ||M|| \left(\frac{||\Delta Y||}{||Y||} + \frac{||\Delta M||}{||M||} \right) . \qquad (12.1.3)$$

All the above equations use the symbol $||\cdot||$, which indicates both a vector norm and its induced matrix norm. When the 2-norm is used, we have

$$||X||_2 \equiv \sqrt{\sum_i |x_i|^2} \; ; \quad ||Y||_2 \equiv \sqrt{\sum_i |y_i|^2}$$

$$||M||_2 \equiv \sup_{||X||_2=1} ||M\,x||_2 \; . \tag{12.1.4}$$

We can, in addition to the above, use the Euclidean norm for M. Let M be complex, of size (n,n), with the moduli of its elements denoted by $|m_{uv}|$, then

$$||M||_E \equiv \left(\sum_{u,v}^{n} |m_{uv}|^2 \right)^{\frac{1}{2}} \; . \tag{12.1.5}$$

Since

$$||M||_2 \leq ||M||_E \leq n^{\frac{1}{2}} ||M||_2$$

$$||M^{-1}||_2 \leq ||M^{-1}||_E \leq n^{\frac{1}{2}} ||M^{-1}||_2 \; , \tag{12.1.6}$$

(12.1.3) implies

$$\frac{||\Delta X||_2}{||X||_2} \leq ||M^{-1}||_E \; ||M||_E \left(\frac{||\Delta Y||_2}{||Y||_2} + \frac{||\Delta M||_2}{||M||_2} \right) \; . \tag{12.1.7}$$

The Turing's N-condition number, N(M), of matrix M, is defined in terms of the norm of M and of its inverse (see for instance Westlake (1968)):

$$N(M) = n^{-1} ||M||_E \; ||M^{-1}||_E \; . \tag{12.1.8}$$

Then (12.1.7) may be written as

$$\frac{||\Delta X||_2}{||X||_2} \leq n\ N(M) \left(\frac{||\Delta Y||_2}{||Y||_2} + \frac{||\Delta M||_2}{||M||_2} \right) . \qquad (12.1.9)$$

N(M) is a measure of the effect on an approximate inverse of matrix M, and a low value of N(M) usually implies a high accuracy of the system's solution. We know that N(M) is always equal to or greater than unity. We would like to have the sequences whose member matrices M have their N(M) not much greater than 1.0, so that the accuracy of inversion is guaranteed.

In this chapter we shall examine the condition numbers of some of the sequences that we have introduced, and show that most of them have a unity condition number.

12.2 Sequence E

In sequence E (see 4.3.4), all the matrix elements have a unity modulus.

$$e_{uv} = \exp(iv\pi(2u-1)/n) \qquad (12.2.1)$$

$$|e_{uv}| \equiv 1 . \qquad (12.2.2)$$

Then

$$||E||_E = \left(\sum_{u,v}^{n} |e_{uv}|^2 \right)^{\frac{1}{2}} = [n^2]^{\frac{1}{2}} = n . \qquad (12.2.3)$$

$$||E^{-1}||_E = ||\hat{E}/n||_E = ||\hat{E}||_E/n = ||E||_E/n = 1. \quad (12.2.4)$$

Therefore

$$N(E) = n^{-1}||E||_E \; ||E^{-1}||_E = 1 \; . \quad (12.2.5)$$

In the case that $R(J_1)$ and $S(J_2)$ belong to the E sequence, then according to (12.2.5)

$$\begin{aligned} N(R) &= 1 \; , \\ N(S) &= 1 \; . \end{aligned} \quad (12.2.6)$$

12.3 Fourier Transform Sequence

As shown in (4.4.1), the sequence has as its matrix elements $\underline{f}_{uv}$ where

$$\underline{f}_{uv} = \exp(-i2\pi uv/n) \quad (12.3.1)$$

and

$$|\underline{f}_{uv}| \equiv 1, \quad |\underline{f}^*_{uv}| \equiv 1. \quad (12.3.2)$$

By (4.4.3),

$$||\underline{F}||_E = \left(\sum_{u,v}^{n} |\underline{F}_{uv}|^2 \right)^{\frac{1}{2}} = n \quad (12.3.3)$$

$$||\underline{F}^{-1}||_E = ||\underline{F}^*/n||_E = ||\underline{F}^*||_E/n = 1$$

$$N(\underline{F}) = n^{-1}||\underline{F}||_E \; ||\underline{F}^{-1}||_E = 1 \; . \quad (12.3.4)$$

In the case that $R(J_1)$ or $S(J_2)$ belong to this $\underline{F}$ sequence,

$$N(R) = 1$$
$$N(S) = 1 . \qquad (12.3.5)$$

12.4 Sequence H

According to (5.1.2)

$$|h_{uv}| = 1 \qquad (12.4.1)$$

Hence, $||H||_E = n$.

The formula for H^{-1} was derived in (5.1.8). Noting that A is merely a permutation matrix and does not affect the norm of H^t,

$$||H^{-1}||_E = ||H^t A/n||_E = ||H^t||_E/n = ||H||_E/n = 1 \qquad (12.4.2)$$

$$N(H) = n^{-1}||H||_E \, ||H^{-1}||_E = 1 . \qquad (12.4.3)$$

12.5 Sequence L

From (5.2.3), L = S ⊗ R has all its elements of unit modulus, if R and S belong to E sequence.

$$||L||_E = \left[\sum_{u,v} |l_{uv}|^2\right]^{\frac{1}{2}} = [J_1^2 J_2^2]^{\frac{1}{2}} = J_1 J_2 . \qquad (12.5.1)$$

From (5.2.4),

$$||L^{-1}||_E = ||R^{-1} \otimes S^{-1}||_E = ||\hat{R} \otimes \hat{S}/J_1 J_2||_E = J_1 J_2 / J_1 J_2 = 1, \qquad (12.5.2)$$

so,

$$N(L) = ||L||_E \, ||L^{-1}||_E / J_1 J_2 = 1 \, . \qquad (12.5.3)$$

In the case that R, S belong to sequence F, (12.5.3) still holds.

12.6 The Vandermonde Sequences

Sequence C is defined in (4.2.2) which gives

$$||C||_E = \left(\sum_{u,v} |x_u^v|^2 \right)^{\frac{1}{2}} \qquad (12.6.1)$$

The inverse of C is given in (4.2.4) and (4.2.5). From (4.2.5) it is seen that

$$\begin{aligned} ||C^{-1}||_E = \{ & [(1+|x_2+x_3+\ldots+x_n|^2 + |x_2x_3+x_3x_4+\ldots+x_{n-1}x_n|^2 + \\ & \ldots+|x_2x_3\ldots x_n|^2)/|x_1(x_2-x_1)(x_3-x_1)\ldots(x_n-x_1)|^2] + \\ & [(1+|x_1+x_3+\ldots+x_n|^2 + |x_1x_3+x_3x_4+\ldots+x_{n-1}x_n|^2 + \\ & \ldots+|x_1x_3\ldots x_n|^2)/|x_2(x_1-x_2)(x_3-x_2)\ldots(x_n-x_2)|^2] + \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ & + [(1+|x_1+x_2+\ldots x_{n-1}|^2 + |x_1x_2+x_2x_3+\ldots+x_{n-2}x_{n-1}|^2 + \\ & \ldots+|x_1x_2\ldots x_{n-1}|^2)/|x_n(x_1-x_n)(x_2-x_n)\ldots|^2]\}^{\frac{1}{2}} \, . \end{aligned} \qquad (12.6.2)$$

and

$$N(C) = n^{-1} ||C||_E \, ||C^{-1}||_E \, . \qquad (12.6.3)$$

(12.6.2) looks complicated. Obviously, the condition number depends on the denominators involving

(x_2-x_1), (x_3-x_1), (x_3-x_2), etc. If an x_k is close to some other x_j, we shall have a very large N(C). This confirms what we claimed earlier: the Vandermonde matrices can easily become unstable.

Fortunately, this thesis does not use the general sequence C. We are only interested in a special case of C, which we called sequence T, defined in (8.4.3). Let us see whether sequence T is unstable.

From (8.2.2) we have

$$t_{uv} = \exp(-iv\Delta zK_{zu}) \tag{12.6.4}$$

$$|t_{uv}| \equiv 1 \ . \tag{12.6.5}$$

Comparing (12.6.4) with (4.2.2) results in

$$x_u = \exp(-i\Delta zK_{zu}) = t_{u1} = t_u \ . \tag{12.6.6}$$

Also $|t_u| \equiv 1$.

From (12.6.1) we shall be able to see the norm of T.

$$||T||_E = \left(\sum_{u,v}^{J_3} |t_{uv}|^2 \right)^{\frac{1}{2}} = [\sum (1)^2]^{\frac{1}{2}} = J_3 \ . \tag{12.6.7}$$

To estimate the norm of T^{-1}, consider the stability requirement stated in Section 9.5. Section 9.5 requires that $t_1, t_2, \ldots, t_{J3}$ make equal angles of $2\pi/J_3$ radians after optimization. Then

$$t_u = \exp(i\ 2\pi\ u/J_3 + i\alpha) \tag{12.6.8}$$

with $\alpha = \Delta zK_{z1} - 2\pi/J_3$.

Because T is of the Vandermonde type $t_{uv}=\exp(i\,2\pi\, uv/J_3+iv\alpha)$. The optimization actually turns T into a Fourier transform matrix given by (4.4.1), post-multiplied by a diagonal rotation matrix D":

$$T = \underline{F}\, D" \quad \text{where } D" = \text{diag}(i\alpha, i2\alpha, \ldots, iv\alpha, \ldots, iJ_3\alpha) \qquad (12.6.9)$$

$$||T^{-1}||_E = ||D"^{-1}\underline{F}^{-1}||_E = ||\underline{F}^{-1}||_E = ||\underline{F}^{*}||_E/J_3$$
$$= ||\underline{F}||_E/J_3 = 1 \,. \qquad (12.6.10)$$

Therefore, the condition number of T is

$$N(T) = J_3^{-1}||T||_E\, ||T^{-1}||_E = J_3^{-1}\cdot J_3\cdot 1 = 1 \,. \qquad (12.6.11)$$

We note that (12.6.11) is based on a matrix T being optimized for $p = p_o$ and $q = q_o$ (reference 9.5.8). In most cases, p and q will be chosen close to p_o and q_o but not equal to them. In these cases we can only expect that T is close to a Fourier transform matrix. Hence N(T) will be near unity but not exactly unity.

12.7 The Error Bound in Solving Pf = F

In this section, we shall estimate the accuracy in f obtained by solving Pf = F using the particular P matrix we created.

Our fast algorithm begins by solving J_1J_2 systems of equations of the form (reference 8.4.4 through 8.4.8)

$$_mT^t{}_mF" = {}_m(\text{Rect}_{J3}F) \quad \text{for } m = 1,2,\ldots,J_1J_2 \,. \qquad (12.7.1)$$

All ${}_mT(J_3)$ are optimized matrices. From (12.6.11), $N({}_mT^t) = N({}_mT)$ are all close to unity. Let us call the maximum condition number N_{max}:

$$N_{max} \equiv \max_m \{N({}_mT)\} \quad . \qquad (12.7.2)$$

Let $||\Delta_m T||_2 = ||\Delta_m T^t||_2$ be the 2-norm of the errors in the computation of the ${}_mT$ matrices. $||\Delta_m T||$ will be due to round-off errors only. It will be small and (12.1.3) applies.

Let $||\Delta_m F''||_2$ and $||\Delta_m F||_2 = ||\Delta_m({}_{Rect_{J3}}F)||_2$ be the 2-norms of the measurement error for the vectors ${}_mF''$ and ${}_m(Rect_{J3}F)$.

According to (12.1.9), these norms are related to one another by

$$||\Delta_m F''||_2 \leq J_3 N({}_mT) \left(\frac{||\Delta_m F||_2}{||{}_mF||_2} + \frac{||\Delta_m T||_2}{||{}_mT||_2} \right) ||{}_mF''||_2 \quad . \qquad (12.7.3)$$

For $m = 1,2,\ldots,J_1J_2$ we shall have J_1J_2 systems of the form (12.7.1), and their corresponding error relations of the form (12.7.3). We shall pick out the one with the maximum error and call the error E_{max}:

$$E_{max} \equiv \max_m \left(\frac{||\Delta_m F||_2}{||{}_mF||_2} + \frac{||\Delta_m T||_2}{||{}_mT||_2} \right) \quad . \qquad (12.7.4)$$

Substituting (12.7.2) and (12.7.4) into (12.7.3), squaring both sides, summing up for $m = 1,2,\ldots,J_1J_2$, taking

square-roots of both sides, and dividing by $||F''||_2$,

$$||\Delta F''||_2/||F''||_2 \leq J_3 N_{max} E_{max} . \qquad (12.7.5)$$

After finding F", our algorithm (8.4.8) proceeds to find f by solving a system of equations Lf = F". From (12.1.9) and (12.7.5),

$$\frac{||\Delta f||_2}{||f||_2} \leq J_1 J_2 N(L) \frac{||\Delta F''||_2}{||F''||_2} \leq J_1 J_2 N(L) J_3 N_{max} E_{max} . \qquad (12.7.7)$$

In the case of (5.2.3), L has a unity condition number. Then

$$\frac{||\Delta f||_2}{||f||_2} \leq J_1 J_2 J_3 N_{max} E_{max} . \qquad (12.7.8)$$

When all ${}_mT$ are ideally optimized, $N_{max} = 1$. Then

$$\frac{||\Delta f||_2}{||f||_2} \leq n E_{max} . \qquad (12.7.9)$$

12.8 Summary

To conclude our error analysis we give the following remarks:

1. Although sequences E and F are Vandermonde sequences, they have a unity condition number, hence they are stable sequences.

2. Whether we force R and S into sequence E, or into sequence F, the result will be the same: $N(R) = 1$, $N(S) = 1$.

3. Sequences L and H have unity condition numbers. They should be ideal for inversion.

4. The matrix T can easily become unstable. However, it can be optimized towards a Fourier transform matrix, in which case it will enjoy a unity condition number.

5. The accuracy of our fast algorithm can be expressed by an error bound

$$\frac{||\Delta f||_2}{||f||_2} \leq n \operatorname*{Max}_m \left(\frac{||\Delta_m F||_2}{||_m F||_2} + \frac{||\Delta_m T||_2}{||_m T||_2} \right) . \qquad (12.8.1)$$

The error bound is n times the maximum relative error in the ${}_mT$ matrices plus n times the maximum relative errors in the data ${}_mF$. It can be interpreted as the N-condition number of P being almost a unity. This is the best P we can possibly create for the system of equations $Pf = P$.

XIII. DISCUSSION

This chapter summarizes our method of solution to inverse scattering problem and the mathematics developed in this thesis. The advantages and disadvantages of our approach as compared with other existing methods will be reviewed.

13.1 Lensless Reconstruction

The conventional way of forming an image of an object through a system of lenses is known to be subject to the following conditions:

(1) The object should be larger than the wavelength of the illuminating radiation.

(2) The aberrations of the lenses are negligible.

(3) A three-dimensional object results only in a two-dimensional image.

The last point is considered as a major handicap of conventional imaging by lenses.

Computational optics studies a wave field by numerical evaluation of field data. It needs no lens and forms no image, and is therefore not subject to the above conditions. With the help of mathematical analysis, the optical data obtained can be processed, and any particular information that enables us to characterize a wave field, for example, a spectrum of spatial

frequencies, can be extracted. The phase information, which no physical instruments of our time can detect directly, has to be sensed by an interferrometric method, like holography or a three-reference-beam method (see Schmidt-Weinmar 1973 and 1975). It is on this basis that computational optics leads to a new dimension in image reconstruction: the lensless reconstruction.

Lensless reconstruction has some advantage over a conventional image forming system in that

(1) the errors due to imperfect lenses are eliminated,

(2) optical messages can be coded and compressed for transmission and can be decoded and restored at a remote distance,

(3) noise can be filtered and distortions can be suppressed so that any optical information contained is enhanced,

(4) a 3-D object can be reconstructed,

(5) information can be extracted concerning waves of a particular type.

This thesis gives an example for points (1), (4) and (5); it shows a possibility of reconstructing a 3-D source density distribution from far-field data.

13.2 Reconstruction of Objects near the Resolution Limit

Two successful approaches have been made previously to 3-D reconstruction. These were computer holography and X-ray crystallography; both are lensless methods.

Holographic techniques are limited to some applications. They require in particular that

(1) The spectral density of the scatterer is band-limited to within a bandwidth of $W \leq 2\pi/8\lambda_o$, where λ_o is the vacuum wavelength of the illuminating light (see, for example, Wolf 1969).

(2) The recording is done in the Fresnel region of the scatterer, and over a plane, not a spherical, surface.

(3) Many sets of data are combined that are to be obtained at one fixed wavelength, and with different directions, of the light that illuminates the object.

A holographic approach will thus not be applicable, if the object to be reconstructed has a microscopic or submicroscopic size. It is usually very difficult to take accurate recordings in the Fresnel region with a photographic film of ordinary thickness.

X-ray crystallography is useful only for structure elucidation of crystals containing molecules of molecular weight less than about 10^5; this is one or two order of magnitude below that of many biologically crucial

molecules such as large nucleic-acid polymers (see Chapline and Wood: X-ray Lasers, Physics Today, June 1975).

The approach developed in this thesis will be combined with the three-reference-beam method to compute, from far-field data, the spatial distribution of the scattered field source density that represents a scatterer. The scattered field source density is computed in discrete form. A density sample at some point in space represents a mean value over the source density within a blocklet inside the scatterer. This is not to assume that the scatterer itself is a discrete object, but to use an approximate representation of the integral over a slowly varying continuous function by a finite number of terms; each of these terms is a weighted average of the integrand in the interval belonging to the particular term. From these samples of source density obtained, we may determine the spatial distribution of scattering potential of the object, i.e. $k_o^2(n^2 - 1)$. This provides a solution to the inverse scattering problem in an area where holographic and X-ray methods cannot be applied.

13.3 Fast Inversion of Large Matrices

A special feature of optical computation is the manipulation of very large matrices, which may represent

either the image of an object or some distribution belonging to a propagated optical field, or some coded message from which an image or the distribution in space of an optical-wave field may be recovered. With the ever increasing demand for better image quality, ever larger matrices have been put in use. Conventional numerical methods are unable to handle such large matrices because of the exceptionally long CPU time and the large core storage required. The propagation of computing errors also presents a problem. Special techniques therefore need to be developed for large matrices.

This thesis has demonstracted a new inversion technique outside the areas of banded, sparse, orthogonal, and ordinary Vandermonde matrices, to which so many contributions have been made. The following results may be noted:

(1) A number of fast methods for direct inversion of certain large matrices have been obtained. The range of matrices, introduced in Chapters II to IV, can be numerically inverted as accurately as, and almost as fast as, orthogonal matrices.

(2) The inversion of certain composite matrices by parts has been shown. For example, a row-wise Kronecker matrix may be inverted, by multiplying

the columns of an inverted factor matrix, with the other constituent matrices inverted.

(3) A technique for solving a system of equations has been given based on rectangularization. An m-plex system (Section 6.3) of equations can be solved quickly, if the data and the unknown vectors are both rectangularized.

The many examples given in Part I indicate that this is an interesting area, worthy of more intensive explorations in future.

13.4 The Method of 3-D Reconstruction

Our method of solution has been based on some physical assumptions:

(1) The object is weakly scattered and its refractive index n varies slowly within the scatterer, i.e., $[\nabla^2(n^2)]/n^2 << 10^{14}$ meter^{-2}.

(2) The scattering object is illuminated by a plane wave field that comes from a monochromatic and coherent source.

(3) The data $F(K_x,K_y)$ are taken from a region far away from the object, i.e. $|X'| >> |X|^2/\lambda_o$, where X' is a space vector in the far-field region, and X is a space vector for any source point that contributes to the far field.

(4) Data $F(K_x,K_y)$ are available at any point $K = (K_x,K_y,K_z)$, with $K_z = (k_o^2 - K_x^2 - K_y^2)^{\frac{1}{2}}$, in the K-domain.

Our 3-D reconstruction is simplified to an inverse mapping of F, the far-field data, onto f, the scattered field source density, or, in other words, to solving a system of equations $Pf = F$ for f. While f remains in the (x,y,z) space domain, F is redefined in the $(\Delta xK_x,\Delta yK_y,\Delta zK_z)$ domain. In view of K_z being dependent on K_x and K_y, F may be considered as defined in the domain's projected plane $(\Delta xK_x,\Delta yK_y)$ for convenience. The indices in the latter domain $k_1,k_2,k_3,\ldots$ are mapped onto a linear index u, $u = 1,2,\ldots,n$. At the same time, the spatial indices j_1,j_2,j_3 are also mapped onto a linear index v, $v = 1,2,\ldots,n$. When these indices are attached to F and f respectively, we have two Hilbert spaces with n-dimensional vectors F and f, and the propagator P maps f onto F. The kernel of P turns out to be separable and symmetric in the x-,y-,z-directions, thus allowing the mapping of P to be performed first in the x-direction, then the y- and finally the z-direction by the operators R, S and T repsectively. To reverse the mapping by P, it is then only necessary to reverse the three directional mappings, one at each time.

The equi-spaced sampling in the (x,y,z) domain, as well as in the projected $(\Delta xK_x,\Delta yK_y)$ plane, yields

redundant factors in matrix P, so that P can be decomposed to allow the application of fast matrix inversions. The selection pattern in the $(\Delta xK_x, \Delta yK_y)$ plane has J_3 rectangular blocks where J_3 is the number of samples in space in the z-direction. The distances between these blocks are equal to multiples of 2π in the directions of either ΔxK_x or ΔyK_y. The blocks are all identical; each block contains a mesh of J_1J_2 points, where J_1 and J_2 are the number of sample points in space in the x- and y-directions.

Our fast algorithms begin by rearranging the data F, then post-multiplying them with T^{-t} (see equations 8.4.7 and 8.4.8). This is the inverse mapping in z-direction, namely, we pick one point in a rectangular block, and the corresponding points in every other block, and operate the data F corresponding to these points with the inverse-map operator T^{-t}. As a result of this operation, we obtain a set of new data F", which are the F's already inverse-mapped in z-direction. Further inverse operations in the other two spatial directions can be done according to the formula $\text{Rect}_{J2}\text{Rect}_{J3}f = (R^{-1})\text{Rect}_{J2}F''(S^{-t})$. The inverse operators in the x- and y-directions are R^{-1} and S^{-t} respectively. The rectangularization of data before each operation, ensures that the points are aligned in the right order.

After these three inverse-operations we shall obtain the required scattered-field source density f, and the equation $Pf = F$ is solved.

13.5 Overall Result of the Method Presented

Our method of solution has been designed to yield an optimum result with regard to the following conditions.

(1) The solution of the linear system $Pf = F$ is speeded up.

(2) R and S are matrices for which a method of fast inversion works.

(3) The Vandermonde matrices T_m, $m = 1,2,\ldots,J_1J_2$, are stabilized.

(4) Computation errors are minimized.

(5) Computer storage requirements are optimized.

Our fast algorithm needs only $J_1J_2(J_1+J_2+J_3+4.5\,J_3^2)$ complex arithmetic operations and approximately $J_1+J_2+J_1J_2J_3$ complex words core storage. This improvement allows us to determine, for example, a $80\times80\times20$ source density representation in 8 minutes on an IBM 360/67 machine without virtual memory. Without such an algorithm, solution of a system of 128,000 equations would not be computationally feasible.

13.6 Proposed Directions for Future Research

In view of the above discussions the following may be suggested for future research as a continuation of the thesis:

(1) To discover other fast-invertible matrices for the categories mentioned in Part I of the thesis

(2) To develop other partial inversion techniques for composite matrices,

(3) An experimental realization of the inverse scattering process to reconstruct the physical structure of a real object,

(4) To compute the optical field near and inside a scatterer given the magnitude and phase of the far field,

(5) Extension of the work towards reconstruction of a submicroscopic object.

BIBLIOGRAPHY

ANDREWS, H.C. Computer Techniques in Image Processing. Academic Press, 73-90, 1970.

ANDREWS, H.C., KANE, J. Kronecker Matrices, Computer Implementation, and Generalized Spectra. J. ACM. 17, No. 2: 260, April 1970.

BJORCK, A., ELFVING, T. Algorithms for Confluent Vandermonde Systems. Numer. Math. 21: 130-137, 1973.

BUTKOV, E. Mathematical Physics. Addison-Wesley Pub. Co., 642, 1968.

CANTIN, G. An Equation Solver of Very Large Capacity. International Journal for Numerical Methods in Engineering 3: 397, 1971.

CARTER, W.H. Computational Reconstruction of Scattering Objects from Hologram. J. Opt. Soc. Amer. 60, No. 3: 306, March 1970.

COOLEY, J.W., TUKEY, J.W. An Algorithm for the Machine Calculation of Complex Fourier Series. Math. Comp. 19: 297, April 1965.

EKSTROM, M.P., A Numerical Algorithm for Identifying Spread Functions of Shift-Invariant Imaging Systems. IEEE Trans.on Computers C-22,No.4: 322-327, April 1973.

FADDEEV, D.K., FADDEEVA, V.N. Computational Methods of Linear Algebra. Translation from Russian by WILLIAMS, R.C., W.H. Freeman Co., 1963.

FORSYTHE, G., MOLER, C.S. Computer Solution of Linear Algebraic Systems. Prentice-Hall Inc., 20-23, 1967.

FRÖBURG, C.E. Introduction to Numerical Analysis. Addison-Wesley Publishing Co., 47-109, 1968.

GABOR, D. Microscopy by Reconstructed Wavefronts. Proc. Phys. Soc. B378: 449-468, 1951.

GOKNAR, I.C. Obtaining the Inverse of the Generalized Vandermonde Matrix of the Most General Type. IEEE Trans. Autom. Control AC-18, No.5: 530-532, Oct. 1973.

GOOD, I.J. The Interactive Algorithm and Practical Fourier Analysis. J. Roy. Stat. Soc. Series B, 361-372, 1958.

GOODMAN, J.W., LAWRENCE, R.W. Digital Image Formation from Electronically Detected Holograms. Appl. Phy. Letter 11, No.3: Aug. 1967.

HICKLING, R. Scattering of Light by Spherical Liquid Droplets Using Computer-Synthesized Holograms. J. Opt. Soc. Amer. 58, No.4: 455-460, April 1968.

ISAACSON, E., KELLER, H.B. Analysis of Numerical Methods. John Wiley & Sons, Inc., 37-39, 1966.

KAN, E.P.F. An Inversion Procedure of the Generalized Vandermonde Matrix. IEEE Trans. Autom. Control AC-16, No. 5: 492, Oct. 1971.

KENDREW, J.C. The 3-Dimensional Structure of a Protein Molecule. Scientific American Offprint 121, 1962.

LESEM, L.B., HIRSCH, P.M., JORDAN, J.A., JR. Computer Synthesis of Holograms for 3-D Display. Comm. ACM. 11, No. 10: 661-674, Oct. 1968.

LESEM, L.B., HIRSCH, P.M., JORDAN, J.A., JR. Computer Generation and Reconstruction of Holograms. Unpublished lecture notes, presented at Symposium on Modern Optics at Polytechnic Inst. of Brooklin, March 1967.

MACON, N., SPITZBARD, A. Inverses of Vandermonde Matrices. Amer. Math. Monthly, 95-100, Feb. 1958.

MALCOLM, M.A., PALMER, J. A Fast Method for Solving a Class of Tridiagonal Linear Systems. Comm. ACM. 17, No.1: 14, Jan. 1974.

McCLELLAN, M.T. The Exact Solution of Systems of Linear Equations with Polynomial Coefficients. J. ACM. 20, No. 4: 563-588, Oct. 1973.

PARKER, F.D. Inverses of Vandermonde Matrices. Amer. Math. Monthly, 410-411, April 1964.

PEREYRA, V., BALLESTER, C. On the Construction of Discrete Approximation to Linear Differential Expression. Math. Comp. 21: 297-302, 1967.

PERUTZ, M.F. The Homoglobin Molecules. Scientific Amer. Nov. 1964.

RALSTON, A. A First Course in Numerical Analysis. McGraw-Hill Book Co., Chapter 9, 1965.

SAYRE, D.S. Computer Technique Aids Protein Analysis. Focal Point, IEEE Spectrum, 28-29, June 1974.

SCHAPPELLE, R.H. The Inverse of the Confluent Vandermonde Matrix. IEEE Trans. Autom. Control AC-17, No. 5: 724-725, Oct. 1972.

SCHMIDT-WEINMAR, H.G. Sampled-Data Solution to the Inverse Problem of Optical Scattering. J. Opt. Soc. Am. 61: 1578, 1971.

SCHMIDT-WEINMAR, H.G. Spatial Distribution of Magnitude and Phase of Optical Wave Fields. J. Opt. Soc. Am. 63, No.5: 547-555, May 1973(1).

SCHMIDT-WEINMAR, H.G. Analysis of the Complex Plane-Wave Spectrum of Microscopic Sources, J. Opt. Soc. Am. 63: 1307, 1973(2).

SCHMIDT-WEINMAR, H.G. Optical-Wave Field Determined over a Hemispherical Boundary Surface, accepted for publication in the Journal of the Optical Society of America: 1975(1).

SCHMIDT-WEINMAR, H.G. Optical-Wave Near Field Specified from Far-Field Data, accepted for publication in the Journal of the Optical Society of America: 1975(2).

STONE, H.S. An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear Systems of Equations. J. ACM. 20, No.1: 27-38, Jan. 1973.

THEILHEIMER, F. A Matrix Version of the Fast Fourier Transform. IEEE Trans. Audio & Electroacoustics AU-17, No.2: 158, June 1969.

TRAUB, J.F. Associated Polynomials and Uniform Methods for the Solution of Linear Problems. SIAM Review 8, No.3: 287, July 1966.

WESTLAKE, J.R. A Handbook of Numerical Matrix Inversion and Solution of Linear Equations. John, Wiley & Sons, 89-116, 1968.

WOLF, E. Determination of the Amplitude and Phase of Scattered Fields by Holography. J. Opt. Soc. Am. 60, No.1: 18, Jan. 1970.

WOLF, E. Three Dimensional Structure Determination of Semi-Transparent Object from Holographic Data. Optics Comm. 1, No. 4: 153-156, Sept.-Oct. 1969.

YOUNG, D.M. On the Solution of Large System of Linear Algebraic Equations with Sparse, Positive Definite Matrices. An article contained in the book by BYRNE, G.D. & HALL, C.A. Numerical Solution of Systems of Non-linear Algebraic Equations. Academic Press, 107-156, 1973.

APPENDIX I

KRONECKER PRODUCT OF MATRICES

Definition A Kronecker multiplication is represented by the operator $\otimes$. Let $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$ be two matrices of size (n,m) and (p,q) respectively. Then the multiplication $A \otimes B$ yields

$$A \otimes B = \{a_{ij}B\}$$

as a Kronecker product or Kronecker matrix of size (np,mq), which is expressible as a partitioned matrix with $[a_{ij}B]$ as the (i,j)th partition, for $i = 1,2,\ldots,m$.

Example

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} & 2\begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} \\ 3\begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} & 4\begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} 5 & 6 & 7 & 10 & 12 & 14 \\ 8 & 9 & 0 & 16 & 18 & 0 \\ 15 & 18 & 21 & 20 & 24 & 28 \\ 24 & 27 & 0 & 32 & 36 & 0 \end{pmatrix}$$

Properties

$$O \otimes A = A \otimes O = O$$

$$(A_1+A_2) \otimes B = (A_1 \otimes B) + (A_2 \otimes B)$$

$$A \otimes (B_1+B_2) = A \otimes B_1 + A \otimes B_2$$

$$pA \otimes qB = pq(A \otimes B)$$

$$A_1A_2 \otimes B_1B_2 = (A_1 \otimes B_1)(A_2 \otimes B_2)$$

$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ if the inverses exist.

Decomposition of a Kronecker Matrix A Kronecker matrix can be decomposed into an ordinary product of matrices which are sparse and of the same size as the original Kronecker matrix. For example,

$$K(6) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 2 & & & & \\ & & 1 & 2 & & \\ & & & & 1 & 2 \\ 3 & 4 & & & & \\ & & 3 & 4 & & \\ & & & & 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & & & \\ & & & 1 & 2 & 3 \\ 4 & 5 & 6 & & & \\ & & & 4 & 5 & 6 \\ 7 & 8 & 9 & & & \\ & & & 7 & 8 & 9 \end{pmatrix}$$

These sparse matrices are usually referred to as Good matrices, and will have regularly repeated elements.

APPENDIX II THE 64 VALUES OF K_u SELECTED BY COMPUTER

$$K_u = (\Delta x K_{xu}\ \Delta y K_{yu}\ \Delta z K_{zu})^t$$

DXKX(1) = 0.78540	DYKY(1) = 0.78540	DZKZ(1) = 62.79018
DXKX(2) = 13.35177	DYKY(2) = 0.78540	DZKZ(2) = 61.35922
DXKX(3) = 0.78540	DYKY(3) = 7.06858	DZKZ(3) = 62.39598
DXKX(4) = 13.35177	DYKY(4) = 7.06858	DZKZ(4) = 60.95576
DXKX(5) = 2.35619	DYKY(5) = 0.78540	DZKZ(5) = 62.75087
DXKX(6) = 14.92256	DYKY(6) = 0.78540	DZKZ(6) = 60.99623
DXKX(7) = 2.35619	DYKY(7) = 7.06858	DZKZ(7) = 62.35643
DXKX(8) = 14.92256	DYKY(8) = 7.06858	DZKZ(8) = 60.59036
DXKX(9) = 3.92699	DYKY(9) = 0.78540	DZKZ(9) = 62.67218
DXKX(10) = 16.49335	DYKY(10) = 0.78540	DZKZ(10) = 60.59038
DXKX(11) = 3.92699	DYKY(11) = 7.06858	DZKZ(11) = 62.27724
DXKX(12) = 16.49335	DYKY(12) = 7.06858	DZKZ(12) = 60.18176
DXKX(13) = 5.49779	DYKY(13) = 0.78540	DZKZ(13) = 62.55396
DXKX(14) = 18.06415	DYKY(14) = 0.78540	DZKZ(14) = 60.14075
DXKX(15) = 5.49779	DYKY(15) = 7.06858	DZKZ(15) = 62.15826
DXKX(16) = 18.06415	DYKY(16) = 7.06858	DZKZ(16) = 59.72908
DXKX(17) = 0.78540	DYKY(17) = 2.35619	DZKZ(17) = 62.75087
DXKX(18) = 13.35177	DYKY(18) = 2.35619	DZKZ(18) = 61.31899
DXKX(19) = 0.78540	DYKY(19) = 8.63938	DZKZ(19) = 62.19795
DXKX(20) = 13.35177	DYKY(20) = 8.63938	DZKZ(20) = 60.75304
DXKX(21) = 2.35619	DYKY(21) = 2.35619	DZKZ(21) = 62.71153
DXKX(22) = 14.92256	DYKY(22) = 2.35619	DZKZ(22) = 60.95576
DXKX(23) = 2.35619	DYKY(23) = 8.63938	DZKZ(23) = 62.15826
DXKX(24) = 14.92256	DYKY(24) = 8.63938	DZKZ(24) = 60.38641
DXKX(25) = 3.92699	DYKY(25) = 2.35619	DZKZ(25) = 62.63280
DXKX(26) = 16.49335	DYKY(26) = 2.35619	DZKZ(26) = 60.54964
DXKX(27) = 3.92699	DYKY(27) = 8.63938	DZKZ(27) = 62.07883
DXKX(28) = 16.49335	DYKY(28) = 8.63938	DZKZ(28) = 59.97643
DXKX(29) = 5.49779	DYKY(29) = 2.35619	DZKZ(29) = 62.51450
DXKX(30) = 18.06415	DYKY(30) = 2.35619	DZKZ(30) = 60.09972
DXKX(31) = 5.49779	DYKY(31) = 8.63938	DZKZ(31) = 61.95947
DXKX(32) = 18.06415	DYKY(32) = 8.63938	DZKZ(32) = 59.52217
DXKX(33) = 0.78540	DYKY(33) = 3.92699	DZKZ(33) = 62.67218
DXKX(34) = 13.35177	DYKY(34) = 3.92699	DZKZ(34) = 61.23846
DXKX(35) = 0.78540	DYKY(35) = 10.21017	DZKZ(35) = 61.95947
DXKX(36) = 13.35177	DYKY(36) = 10.21017	DZKZ(36) = 60.50887
DXKX(37) = 2.35619	DYKY(37) = 3.92699	DZKZ(37) = 62.63280
DXKX(38) = 14.92256	DYKY(38) = 3.92699	DZKZ(38) = 60.87476
DXKX(39) = 2.35619	DYKY(39) = 10.21017	DZKZ(39) = 61.91963
DXKX(40) = 14.92256	DYKY(40) = 10.21017	DZKZ(40) = 60.14075
DXKX(41) = 3.92699	DYKY(41) = 3.92699	DZKZ(41) = 62.55396

DXKX(42) = 16.49335	DYKY(42) = 3.92699	DZKZ(42) = 60.46808
DXKX(43) = 3.92699	DYKY(43) = 10.21017	DZKZ(43) = 61.83989
DXKX(44) = 16.49335	DYKY(44) = 10.21017	DZKZ(44) = 59.72908
DXKX(45) = 5.49779	DYKY(45) = 3.92699	DZKZ(45) = 62.43552
DXKX(46) = 18.06415	DYKY(46) = 3.92699	DZKZ(46) = 60.01755
DXKX(47) = 5.49779	DYKY(47) = 10.21017	DZKZ(47) = 61.72006
DXKX(48) = 18.06415	DYKY(48) = 10.21017	DZKZ(48) = 59.27292
DXKX(49) = 0.78540	DYKY(49) = 5.49779	DZKZ(49) = 62.55396
DXKX(50) = 13.35177	DYKY(50) = 5.49779	DZKZ(50) = 61.11746
DXKX(51) = 0.78540	DYKY(51) = 11.78097	DZKZ(51) = 61.68008
DXKX(52) = 13.35177	DYKY(52) = 11.78097	DZKZ(52) = 60.22275
DXKX(53) = 2.35619	DYKY(53) = 5.49779	DZKZ(53) = 62.51450
DXKX(54) = 14.92256	DYKY(54) = 5.49779	DZKZ(54) = 60.75304
DXKX(55) = 2.35619	DYKY(55) = 11.78097	DZKZ(55) = 61.64006
DXKX(56) = 14.92256	DYKY(56) = 11.78097	DZKZ(56) = 59.85287
DXKX(57) = 3.92699	DYKY(57) = 5.49779	DZKZ(57) = 62.43552
DXKX(58) = 16.49335	DYKY(58) = 5.49779	DZKZ(58) = 60.34554
DXKX(59) = 3.92699	DYKY(59) = 11.78097	DZKZ(59) = 61.55995
DXKX(60) = 16.49335	DYKY(60) = 11.78097	DZKZ(60) = 59.43921
DXKX(61) = 5.49779	DYKY(61) = 5.49779	DZKZ(61) = 62.31685
DXKX(62) = 18.06415	DYKY(62) = 5.49779	DZKZ(62) = 59.89409
DXKX(63) = 5.49779	DYKY(63) = 11.78097	DZKZ(63) = 61.43959
DXKX(64) = 18.06415	DYKY(64) = 11.78097	DZKZ(64) = 58.98080

APPENDIX III

A. A Computer Program to Generate Matrices R, S, T, and Data F, and Store the Outputs in File DATA.

```
                C     ********* PROGRAM TO GENERATE DATA ********
                C
   ISN 0002             COMPLEX R(4,4),T(16,4,4)
   ISN 0003             COMPLEX CMPLX,F(16), BIGF(64)
   ISN 0004             COMMON NX,NY,NZ,TWOPI
   ISN 0005             READ (5,20) NX,NY,NZ,OK
   ISN 0006      20     FORMAT (3I2,F5.2)
   ISN 0007             NXY=NX*NY
   ISN 0008             NXYZ=NXY*NZ
   ISN 0009             TWOPI = 6.2831853
                C
                C     COMPUTER SIMULATED DATA
                C     ACTUAL PROBLEM BIGF WILL BE MEASURED.
                C     MATRICES R,S,T WILL BE COMPUTER GENERATED.
                C     BIGF IS NOW COMPUTER GENERATED.
   ISN 0010             CALL GENR(R,NX)
   ISN 0011             WRITE(6,150)
   ISN 0012      150    FORMAT ('1')
   ISN 0013             DO 170 I=1,4
   ISN 0014      170    WRITE (6,180)
   ISN 0015      180    FORMAT ('-')
   ISN 0016             CALL GENT(T,NXY,OK)
   ISN 0017             WRITE(6,150)
   ISN 0018             CALL GENBF(BIGF,R,T)
   ISN 0019             WRITE(7,50) ((R(J1,J),J=1,NX),J1=1,NX)
   ISN 0020             WRITE (7,50)   (((T(K1,K2,K3),K3=1,NZ),K2=1,
                       .NZ),K1=1,NXY)
   ISN 0021             WRITE(7, 50) (BIGF(I),I=1,NXYZ)
   ISN 0022      50     FORMAT    (4(F15.7,F15.7) )
   ISN 0023             STOP
   ISN 0024             END

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =      23 ,PROGRAM SIZE =     3434

*STATISTICS*  NO DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME       2.620 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                     C
   ISN 0002                SUBROUTINE GENR(R,NN)
                     C     GENERATION OF MATRIX R & S, R=S IN THIS CASE.
   ISN 0003                COMPLEX R(NN,NN),CMPLX
   ISN 0004                A=3.141593 /NN
   ISN 0005                DO 10 J1=1,NN
   ISN 0006                JJ=(2*J1-1)
   ISN 0007                DO 10 J=1,NN
   ISN 0008                COSA = COS(A*JJ*J)
   ISN 0009                IF (COSA.LT.-0.999999) COSA=-1.
   ISN 0011                IF (COSA.GT.0.999999) COSA=1.
   ISN 0013                IF((COSA.LT.0.000002).AND.(COSA.GT.-0.000002
                          .)) COSA=0.0000
   ISN 0015                SINA =-SIN(A*JJ*J)
   ISN 0016                IF (SINA.GT.0.999999) SINA=1.
   ISN 0018                IF((SINA.LT.0.000002).AND.(SINA.GT.-0.000002
                          .)) SINA=0.0000
   ISN 0020                IF (SINA.LT.-0.999999) SINA=-1.
   ISN 0022          10    R(J1,J) = CMPLX(COSA,SINA)
   ISN 0023                WRITE (6,150)
   ISN 0024          150   FORMAT ('1')
   ISN 0025                DO 170 I=1,4
   ISN 0026          170   WRITE (6,180)
   ISN 0027          180   FORMAT ('-')
   ISN 0028                WRITE(6,30) ((R(J1,J),J=1,NN),J1=1,NN)
   ISN 0029          30    FORMAT('0R=',4(4(F7.4,F7.4,1X)/,'    '))
   ISN 0030                RETURN
   ISN 0031                END

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,

*OPTICNS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*   SOURCE STATEMENTS =      30 ,PROGRAM SIZE =     1090

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END CF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME       1.796 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)       (CT204)

        COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                           SOURCE,EBCDIC,NOLIST,DECK,NCLOAD,NOMAP

 ISN 0002                SUBROUTINE GENT(T,NXY,OK)
                    C    GENERATION OF MATRIX T.
 ISN 0003                COMMON NX,NY,NZ,TWOPI
 ISN 0004                COMPLEX T(NXY,NZ,NZ), CMPLX
 ISN 0005                OKSQ=OK*OK
 ISN 0006                NZD2 = NZ/2.
 ISN 0007                A=3.141593 /NX*2.
 ISN 0008                B=3.141593 /NX*2.
                    C    FOR SAKE OF SIMPLICITY DXKX IS WRITTEN AS XK,
                    C    DYKY AS YK, DZKZ AS ZK.
 ISN 0009                YK1=-3.141593 /NY-6.2831853
 ISN 0010                XKO=-3.141593 /NX-12.56637
 ISN 0011                N=0
 ISN 0012                NNT = 0
 ISN 0013                DO 110 K2=1,NY
 ISN 0014                YK1=YK1+B
 ISN 0015                XK1=XKO
 ISN 0016                DO 110 K1=1,NX
 ISN 0017                XK1=XK1+A
 ISN 0018                NNT= NNT +1
 ISN 0019                NT=0
 ISN 0020                YK=YK1
 ISN 0021                DO 110 K4=1,2
 ISN 0022                YK=YK+6.283185
 ISN 0023                XK=XK1
 ISN 0024                DO 110 K3=1,NZD2
 ISN 0025                XK=XK+12.56637
 ISN 0026                NT=NT+1
 ISN 0027                N=N+1
                    C    ASSUMING DX=DY=DZ=1 MICRON.
                    C    ACTUAL FORMULA IS .***   DZKZ =
                    C    DZ*(SQRT(OKSQ-(DXKX/DX)**2-(DYKY/DY)**2))
 ISN 0028                ZK=SQRT(OKSQ-XK*XK-YK*YK)
 ISN 0029                WRITE (6,120) N  ,XK,N  ,YK,N  ,ZK
 ISN 0030           120  FORMAT('    DXKX(',I2,')=',F9.5,'   DYKY(',
                        .I2,')=',F9.5,'   DZKZ(',I2,')=',F9.5)
 ISN 0031                DO 130 K=1,NZ
 ISN 0032                COSA = COS(ZK*K)
 ISN 0033                SINA = -SIN(ZK*K)
 ISN 0034           130  T(NNT, NT,K)  = CMPLX ( COSA, SINA )
 ISN 0035           110  CONTINUE
 ISN 0036                WRITE(6,150)
 ISN 0037           150  FORMAT ('1')
 ISN 0038                DO 170 I=1,4
 ISN 0039           170  WRITE (6,180)
 ISN 0040           180  FORMAT ('-')
 ISN 0041                DO 160 K1=1,6
 ISN 0042           160  WRITE(6,140) ((T(K1,K2,K3),K3=1,NZ),K2=1,NZ)
```

```
ISN 0043      140   FORMAT('0T=',4(4(F7.4,F7.4,1X)/,'    '))
ISN 0044            WRITE(6,150)
ISN 0045            DO 200 I=1,2
ISN 0046      200   WRITE (6,180)
ISN 0047            DO 190 K1= 7,NXY
```

```
   ISN 0048      190    WRITE(6,140) ((T(K1,K2,K3),K3=1,NZ),K2=1,NZ)
   ISN 0049             RETURN
   ISN 0050             END

*OPTIONS IN EFFECT*       NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*       SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =      49 ,PROGRAM SIZE =     1792

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME       4.626 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NCLOAD,NOMAP

  ISN 0002            SUBROUTINE GENBF(BIGF,R,T)
                C     A ROUTINE TO SIMULATE THE DATA BIGF
                C     FOR SAKE OF SIMPLICITY, A SCATTERING OBJECT
                C     IS REPRESENTED BY SCATTERING POTENTIALS AT
                C     POINTS: (2,1,3), (2,4,3), (2,1,4), (2,4,4),
                C     (3,1,3), (3,4,3), (3,1,4), (3,4,4) WHERE
                C     SMALLF = 2+I, I**2=-1, OTHERWISE SMALLF = 0.
  ISN 0003            COMMON NX,NY,NZ,TWOPI
  ISN 0004            DIMENSION L1(8),L2(8),L3(8)
  ISN 0005            COMPLEX BIGF(64),SMALLF
  ISN 0006            COMPLEX R(NX,NX),CMPLX,T(16,NZ,NZ)
  ISN 0007            DO 1 I=1,4
  ISN 0008       1    WRITE (6,2)
  ISN 0009       2    FORMAT ('-')
  ISN 0010            NXYZ = NX*NY*NZ
  ISN 0011            N=0
  ISN 0012            SMALLF = CMPLX(2.,1.)
  ISN 0013            DO 400 I=1,8
  ISN 0014     400    READ 410, L1(I),L2(I),L3(I)
  ISN 0015     410    FORMAT (3I1)
  ISN 0016     420    N=N+1
  ISN 0017            BIGF(N) = CMPLX (0.,0.)
  ISN 0018            NNT =(N-1)/NZ+1
  ISN 0019            NR = MOD((N-1)/NZ,NX) + 1
  ISN 0020            NS = (N-1)/(NX*NZ) + 1
  ISN 0021            NT = MOD(N-1,NZ)+1
  ISN 0022            DO 430 I=1,8
  ISN 0023     430    BIGF(N)=BIGF(N)+R(NR,L1(I))*R(NS,L2(I))*T(
                     .NNT,NT,L3(I))*SMALLF
  ISN 0024            IF (N.GE.NXYZ) GO TO 440
  ISN 0026            GO TO 420
  ISN 0027     440    DO 450 I=1,NXYZ
  ISN 0028     450    WRITE(6,451) I,BIGF(I)
  ISN 0029     451    FORMAT( 10X,'   BIGF(',I2,')=',F15.7,F15.7/)
  ISN 0030            RETURN
  ISN 0031            END

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =      30 ,PROGRAM SIZE =     1242

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME       1.756 SEC.
```

The Matrix R(4) or S(4) Generated by Computer

```
R= 0.7071-0.7071  0.0   -1.0000 -0.7071-0.7071 -1.0000 0.0
  -0.7071-0.7071  0.0    1.0000  0.7071-0.7071 -1.0000 0.0
  -0.7071 0.7071  0.0   -1.0000  0.7071 0.7071 -1.0000 0.0
   0.7071 0.7071  0.0    1.0000 -0.7071 0.7071 -1.0000 0.0
```

The 16 Matrices T(4) Generated by Computer

```
T= 0.9991 0.0417  0.9965 0.0833  0.9922 0.1247  0.9861 0.1659
   0.0980 0.9952 -0.9808 0.1951 -0.2903-0.9569  0.9239-0.3826
   0.9065 0.4222  0.6435 0.7655  0.2602 0.9656 -0.1718 0.9851
  -0.3006 0.9538 -0.8193-0.5733  0.7931-0.6091  0.3425 0.9395

T= 0.9967 0.0809  0.9869 0.1613  0.9706 0.2406  0.9480 0.3183
  -0.2617 0.9651 -0.8630-0.5052  0.7135-0.7007  0.4895 0.8720
   0.8891 0.4577  0.5810 0.8139  0.1440 0.9896 -0.3249 0.9458
  -0.6215 0.7834 -0.2274-0.9738  0.9042 0.4271 -0.8966 0.4429

T= 0.9873 0.1590  0.9494 0.3139  0.8874 0.4609  0.8029 0.5961
  -0.6215 0.7834 -0.2274-0.9738  0.9042 0.4271 -0.8966 0.4429
   0.8501 0.5266  0.4453 0.8954 -0.0929 0.9957 -0.6033 0.7975
  -0.8816 0.4720  0.5545-0.8322 -0.0961 0.9954 -0.3850-0.9229

T= 0.9616 0.2743  0.8495 0.5276  0.6721 0.7404  0.4432 0.8964
  -0.9002 0.4354  0.6208-0.7839 -0.2176 0.9760 -0.2291-0.9734
   0.7816 0.6238  0.2218 0.9751 -0.4349 0.9005 -0.9016 0.4325
  -0.9992 0.0388  0.9970-0.0776 -0.9932 0.1162  0.9880-0.1547

T= 0.9967 0.0809  0.9869 0.1613  0.9706 0.2406  0.9480 0.3183
   0.0579 0.9983 -0.9933 0.1156 -0.1729-0.9849  0.9733-0.2296
   0.8057 0.5923  0.2984 0.9544 -0.3249 0.9457 -0.8219 0.5696
  -0.4864 0.8737 -0.5267-0.8500  0.9989-0.0467 -0.4451 0.8955

T= 0.9928 0.1200  0.9712 0.2383  0.9356 0.3532  0.8864 0.4629
  -0.3006 0.9538 -0.8193-0.5733  0.7931-0.6091  0.3425 0.9395
   0.7816 0.6238  0.2218 0.9751 -0.4349 0.9005 -0.9016 0.4325
  -0.7673 0.6413  0.1775-0.9841  0.4949 0.8690 -0.9370-0.3494
```

```
T= 0.9803 0.1977  0.9218 0.3877  0.8269 0.5623  0.6994 0.7147
  -0.6529 0.7574 -0.1474-0.9891  0.8454 0.5341 -0.9565 0.2916
   0.7296 0.6838  0.0647 0.9979 -0.6352 0.7723 -0.9916 0.1291
  -0.9593 0.2823  0.8406-0.5416 -0.6536 0.7569  0.4134-0.9106

T= 0.9501 0.3121  0.8052 0.5929  0.5800 0.8146  0.2968 0.9549
  -0.9173 0.3981  0.6830-0.7304 -0.3358 0.9419 -0.0670-0.9978
   0.6430 0.7659 -0.1731 0.9849 -0.8656 0.5007 -0.9401-0.3410
  -0.9859-0.1673  0.9440 0.3299 -0.8755-0.4832  0.7824 0.6228

T= 0.9873 0.1590  0.9494 0.3139  0.8874 0.4609  0.8029 0.5961
  -0.0226 0.9997 -0.9990-0.0452  0.0677-0.9977  0.9959 0.0902
   0.6430 0.7659 -0.1731 0.9849 -0.8656 0.5007 -0.9401-0.3410
  -0.6832 0.7302 -0.0664-0.9978  0.7739 0.6333 -0.9912 0.1324

T= 0.9803 0.1977  0.9218 0.3877  0.8269 0.5623  0.6994 0.7147
  -0.3768 0.9263 -0.7161-0.6980  0.9164-0.4003  0.0256 0.9997
   0.6120 0.7909 -0.2509 0.9680 -0.9191 0.3940 -0.8741-0.4858
  -0.9002 0.4354  0.6208-0.7839 -0.2176 0.9760 -0.2291-0.9734

T= 0.9616 0.2743  0.8495 0.5276  0.6721 0.7404  0.4432 0.8964
  -0.7124 0.7017  0.0152-0.9999  0.6908 0.7230 -0.9995-0.0303
   0.5470 0.8371 -0.4015 0.9159 -0.9863 0.1649 -0.6776-0.7354
  -0.9992 0.0388  0.9970-0.0776 -0.9932 0.1162  0.9880-0.1547

T= 0.9225 0.3860  0.7019 0.7122  0.3726 0.9280 -0.0146 0.9999
  -0.9469 0.3215  0.7933-0.6088 -0.5555 0.8315  0.2587-0.9660
   0.4431 0.8965 -0.6074 0.7944 -0.9813-0.1926 -0.2621-0.9650
  -0.9142-0.4053  0.6714 0.7411 -0.3134-0.9496 -0.0984 0.9951

T= 0.9616 0.2743  0.8495 0.5276  0.6721 0.7404  0.4432 0.8964
  -0.1431 0.9897 -0.9590-0.2833  0.4176-0.9086  0.8395 0.5433
   0.4069 0.9135 -0.6689 0.7433 -0.9512-0.3086 -0.1051-0.9945
  -0.8615 0.5077  0.4845-0.8748  0.0267 0.9996 -0.5305-0.8477

T= 0.9501 0.3121  0.8052 0.5929  0.5800 0.8146  0.2968 0.9549
  -0.4864 0.8737 -0.5267-0.8500  0.9989-0.0467 -0.4451 0.8955
   0.3700 0.9290 -0.7262 0.6875 -0.9074-0.4203  0.0548-0.9985
  -0.9868 0.1619  0.9476-0.3195 -0.8833 0.4687  0.7958-0.6056
```

```
T= 0.9225 0.3860  0.7019 0.7122  0.3726 0.9280 -0.0146 0.9999
  -0.7929 0.6094  0.2573-0.9663  0.3848 0.9230 -0.8676-0.4973
   0.2945 0.9557 -0.8266 0.5628 -0.7813-0.6242  0.3665-0.9304
  -0.9687-0.2484  0.8766 0.4813 -0.7295-0.6839  0.5368 0.8437

T= 0.8703 0.4925  0.5148 0.8573  0.0258 0.9997 -0.4699 0.8827
  -0.9793 0.2024  0.9181-0.3965 -0.8188 0.5741  0.6856-0.7279
   0.1776 0.9841 -0.9369 0.3495 -0.5104-0.8600  0.7557-0.6550
  -0.7587-0.6514  0.1513 0.9885  0.5291-0.8485 -0.9542 0.2991
```

The 64 Values of F_u Generated by Computer

BIGF(1)=	-6.2256298	1.1064100
BIGF(2)=	-0.5613050	4.6524277
BIGF(3)=	-2.2156420	-5.7637615
BIGF(4)=	-3.7373371	0.1444349
BIGF(5)=	1.7786541	-1.9204302
BIGF(6)=	0.9054432	-1.3089981
BIGF(7)=	2.1614275	1.3455830
BIGF(8)=	1.0237227	0.5006943
BIGF(9)=	0.6133854	-2.5383072
BIGF(10)=	1.0779219	-0.3698998
BIGF(11)=	2.5166521	0.1224317
BIGF(12)=	0.2893295	0.5678787
BIGF(13)=	0.8399563	6.2070236
BIGF(14)=	-1.2671318	-0.6242361
BIGF(15)=	-5.6654854	1.8798981
BIGF(16)=	0.0395203	-0.1162157
BIGF(17)=	-10.3667536	11.1930933
BIGF(18)=	5.5610733	9.6120625
BIGF(19)=	-6.4317226	-13.0047417
BIGF(20)=	-7.6865349	0.8837890
BIGF(21)=	0.6263094	-6.2819653

BIGF(22) =	-0.1444349	-3.7373381
BIGF(23) =	5.3353882	2.6768169
BIGF(24) =	2.0007019	0.8067846
BIGF(25) =	-2.5111609	-5.7705317
BIGF(26) =	1.4820700	-2.1783590
BIGF(27) =	5.8762512	-0.2490253
BIGF(28) =	0.4620686	0.7744809
BIGF(29) =	10.6225891	10.6993713
BIGF(30) =	-3.0517349	0.5681009
BIGF(31) =	-12.6921244	5.5165777
BIGF(32) =	-0.3376341	1.2364845
BIGF(33) =	-3.5750589	14.7943602
BIGF(34) =	9.1402493	5.5127983
BIGF(35) =	-5.5165701	-12.6921263
BIGF(36) =	-6.0702972	0.2750196
BIGF(37) =	-2.5111656	-5.7705383
BIGF(38) =	-1.8166704	-3.0272770
BIGF(39) =	4.9420080	2.7957745
BIGF(40) =	1.2671318	0.6242361
BIGF(41) =	-4.9829588	-3.7950935
BIGF(42) =	0.2333988	-2.3867531
BIGF(43) =	5.5622482	0.0474453
BIGF(44) =	0.0542316	0.1101189
BIGF(45) =	14.3868771	4.1374674
BIGF(46) =	-1.9313088	1.5677176
BIGF(47) =	-12.1498728	4.5382385
BIGF(48) =	-1.0951471	2.9674492

BIGF(49) =	0.8399525	6.2070198
BIGF(50) =	4.0722971	0.7445297
BIGF(51) =	-1.1449089	-5.1794376
BIGF(52) =	-1.6398525	-0.2827377
BIGF(53) =	-1.8225460	-1.8357210
BIGF(54) =	-1.0397530	-0.8253107
BIGF(55) =	1.6471977	1.4098988
BIGF(56) =	0.1654558	0.1337700
BIGF(57) =	-2.4683943	-0.7098782
BIGF(58) =	-0.2159306	-0.8149223
BIGF(59) =	2.0643368	0.4247271
BIGF(60) =	-0.0742826	-0.3194571
BIGF(61) =	6.0912542	-0.5498953
BIGF(62) =	-0.3284135	0.5533237
BIGF(63) =	-4.7900763	0.7790203
BIGF(64) =	-1.2231808	1.8246908

B. A Source Program to Solve Pf = F

This program is to be compiled under *FORTG and the object program will be stored in File COMPUT for later use.

```
                C     ********* MAIN PROGRAM ********
ISN 0002              COMPLEX R(4,4),T(16,4,4),U(16,4),V(4,4)
ISN 0003              COMPLEX CMPLX,F(16), BIGF(64)
ISN 0004              COMPLEX M(4,4),Q(4),SMALLF(64)
ISN 0005              COMMON NX,NY,NZ,TWOPI
ISN 0006              NX=4
ISN 0007              NY=4
ISN 0008              NZ=4
ISN 0009              NXY=NX*NY
ISN 0010              NXYZ=NXY*NZ
ISN 0011              TWOPI = 6.2831853
ISN 0012              READ (7,50) ((R(J1,J),J=1,NX),J1=1,NX)
ISN 0013        50    FORMAT     (4(F15.7,F15.7) )
ISN 0014              READ  (7,50)   (((T(K1,K2,K3),K3=1,NZ),K2=1,
                     .NZ),K1=1,NXY)
ISN 0015              READ (7, 50) (BIGF(I),I=1,NXYZ)
                C     SOLUTION OF PF=BIGF FOR F, USING FAST ALGORM.
                C     AS A FIRST STEP, MULTIPLY BIGF WITH TM1.
ISN 0016              NF =0
ISN 0017              DO 330 I=1,NXY
ISN 0018              CALL TRANSF(BIGF,NF,F,NZ)
ISN 0019              DO 310 J=1,NZ
ISN 0020              DO 310 K=1,NZ
ISN 0021              V(J,K) = T(I,J,K)
ISN 0022        310   CONTINUE
ISN 0023              CALL INVV (V,NZ,M)
ISN 0024              CALL MATMLT(M,F,Q,NZ,NZ,1)
ISN 0025              DO 320 J=1,NZ
                C     TRANSPOSING U
ISN 0026        320   U(I,J) = Q(J)
ISN 0027        330   CONTINUE
                C     SOLVING SYSTEM COLUMN BY COLUMN
ISN 0028              NF=0
ISN 0029              DO 350 I=1,NZ
ISN 0030              DO 340 J=1,NXY
ISN 0031        340   F(J) = U(J,I)
ISN 0032              CALL LAMSCL(F,NXY,R,SMALLF,NF)
ISN 0033        350   CONTINUE
ISN 0034              WRITE(6,150)
ISN 0035        150   FORMAT ('1')
ISN 0036              CALL PROUT(SMALLF,NXYZ,1)
ISN 0037              STOP
ISN 0038              END
```

```
*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,
*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP
*STATISTICS*     SOURCE STATEMENTS =      37 ,PROGRAM SIZE =     5122
*STATISTICS*  NO  DIAGNOSTICS GENERATED
****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME      3.656 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                C
  ISN 0002            SUBROUTINE TRANSF(BIGF,NF,F,NZ)
                C     A ROUTINE TO TRANSFER THE ELEMENTS OF BIGF,
                C     FROM (NF+1)TH TO (NF+NZ)TH, TO THE F(NZ).
  ISN 0003            COMPLEX F(16),BIGF(64)
  ISN 0004            DO 10 I=1,NZ
  ISN 0005            NF = NF+1
  ISN 0006     10     F(I) = BIGF(NF)
  ISN 0007            RETURN
  ISN 0008            END

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =       7 ,PROGRAM SIZE =      356

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME       .726 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP


                    C
  ISN 0002                  SUBROUTINE INVE(E,N,Q)
                    C       A ROUTINE TO INVERT A SPECIAL MATRIX E.
                    C       HERE Q IS EQUAL TO E ROTATED CLOCKWISE.
  ISN 0003                  COMPLEX E(N,N),Q(N,N)
  ISN 0004                  DO 50 J3=1,N
  ISN 0005                  DO 50 J=1,N
  ISN 0006          50      Q(J,J3)=E(N -J3+1,J)/N
  ISN 0007                  RETURN
  ISN 0008                  END

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =       7 ,PROGRAM SIZE =      562

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME        .623 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

        COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                           SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                C
  ISN 0002              SUBROUTINE MATMLT(A,B,C,M,N,L)
                C       THE SUBROUTINE WILL COMPUTE MATRIX C(M,L) AS
                C       A PRODUCT OF A(M,N) AND B(N,L).
  ISN 0003              COMPLEX   A(M,N),B(N,L),C(M,L)
  ISN 0004              COMPLEX CMPLX
  ISN 0005              DO 10 I=1,M
  ISN 0006              DO 10 J=1,L
  ISN 0007              C(I,J)=CMPLX(0.,0.)
  ISN 0008              DO 10 K=1,N
  ISN 0009     10       C(I,J)=C(I,J)+A(I,K)*B(K,J)
  ISN 0010              RETURN
  ISN 0011              END

*OPTIONS IN EFFECT*     NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*     SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =      10 ,PROGRAM SIZE =      648

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME        .653 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)     (CT204)

       / COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                            SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                 C
  ISN 0002               SUBROUTINE PROUT(C,M,L)
                 C       THE SUBROUTINE WILL OUTPUT THE MATRIX C
  ISN 0003               COMPLEX   C(M,L)
  ISN 0004               DO 1 I=1,4
  ISN 0005       1       WRITE (6,2)
  ISN 0006       2       FORMAT ('-')
  ISN 0007               DO 10 I=1,M
  ISN 0008       10      PRINT 20, I,(C(I,J),J=1,L)
  ISN 0009       20      FORMAT(10X,'     F(',I2,')=',F15.7,F15.7/)
  ISN 0010               RETURN
  ISN 0011               END

*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =      10 ,PROGRAM SIZE =      434

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

    COMPILER STATISTICS:   ELAPSED TIME        .786 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)       (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                C
 ISN 0002             SUBROUTINE LAMSOL(Y,NXY,R,X,N)
                C     A ROUTINE TO COMPUTE F=(RM1)(U)(SM1T)
 ISN 0003             COMMON NX,NY,NZ,TWOPI
 ISN 0004             COMPLEX  R(4,4),Q(4,4),SM1T(4,4)
 ISN 0005             COMPLEX*Y(NXY),M(4,4),F(4,4),U(4,4),X(64)
 ISN 0006             CALL INVE(R,NX,Q)
 ISN 0007             CALL ARANGE(Y,NXY,U,NX,NY)
 ISN 0008             CALL INTRAN(R,NX,SM1T)
 ISN 0009             CALL MATMLT(U,SM1T,M,NX,NY,NY)
 ISN 0010             CALL MATMIT(Q,M,F,NX,NX,NY)
 ISN 0011             DO 10 J=1,NY
 ISN 0012             DO 10 I=1,NX
 ISN 0013             N=N+1
 ISN 0014      10     X(N)=F(I,J)
 ISN 0015             RETURN
 ISN 0016             END

*OPTIONS IN EFFECT*       NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*       SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =      15 ,PROGRAM SIZE =     1252

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME        .806 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)        (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                  C
    ISN 0002            SUBROUTINE ARANGE(X,N,XPRIME,NBYJ,J)
                  C     A ROUTINE TO REARRANGE A (N,1) MATRIX TO
                  C     FORM A(N/J,J) MATRIX.
    ISN 0003            COMPLEX X(N)   ,XPRIME(NBYJ,J)
    ISN 0004            I=0.
    ISN 0005            DO 10 NN=1,J
    ISN 0006            DO 10 MM=1,NBYJ
    ISN 0007            I=I+1
    ISN 0008       10   XPRIME(MM,NN)=X(I)
    ISN 0009            RETURN
    ISN 0010            END

*OPTICNS IN EFFECT*       NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*       SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =        9 ,PROGRAM SIZE =      410

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME        .543 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

          COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                             SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

                    C
 ISN 0002           C     SUBROUTINE INTRAN(E,N,Q)
                    C     INVERT AND TRANSPOSE MATRIX E, OR TURN E UP
                    C     SIDE DOWN
 ISN 0003                 COMPLEX E(N,N),Q(N,N)
 ISN 0004                 DO 50 J3=1,N
 ISN 0005                 DO 50 J=1,N
 ISN 0006           50    Q(J,J3) = E(N+1-J,J3)/N
 ISN 0007                 RETURN
 ISN 0008                 END

*OPTIONS IN EFFECT*       NAME=  MAIN,OPT=01,LINECNT=59,

*OPTIONS IN EFFECT*       SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

*STATISTICS*     SOURCE STATEMENTS =        7 ,PROGRAM SIZE =      562

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******

     COMPILER STATISTICS:   ELAPSED TIME        .770 SEC.
```

```
MTS FORTRAN H (OS REL 21.7)      (CT204)

        COMPILER OPTIONS - NAME=  MAIN,OPT=01,LINECNT=59,
                           SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP

ISN 0002            SUBROUTINE INVV(AA,N,AINV)
                C   A ROUTINE TO INVERT VANDERMONDE MATRIX
ISN 0003            COMPLEX A,AINV,AA,W
ISN 0004            COMPLEX CMPLX
ISN 0005            DIMENSION AA(N,N),AINV(N,N),A(20,40),ID(20)
ISN 0006            NN=N+1
ISN 0007            N2=2*N
ISN 0008            DO 100 I=1,N
ISN 0009            ID(I)=I
ISN 0010            DO 100 J=1,N
ISN 0011        100 A(I,J)=AA(I,J)
ISN 0012            DO 200 I=1,N
ISN 0013            DO 200 J=NN,N2
ISN 0014        200 A(I,J)=0.
ISN 0015            DO 300 I=1,N
ISN 0016        300 A(I,N+I)=1.
ISN 0017            K=1
ISN 0018          1 CONTINUE
ISN 0019          2 IF(CABS(A(K,K))) 3,999,3
ISN 0020          3 KK=K+1
ISN 0021            DO 4 J=KK,N2
ISN 0022            A(K,J)=A(K,J)/A(K,K)
ISN 0023            DO 4 I=1,N
ISN 0024            IF(K-I) 41,4,41
ISN 0025         41 W=A(I,K)*A(K,J)
ISN 0026            A(I,J)=A(I,J)-W
ISN 0027            IF (CABS(A(I,J))-.0001*CABS(W)) 42,4,4
ISN 0028         42 A(I,J)=CMPLX(0.,0.)
ISN 0029          4 CONTINUE
ISN 0030            K=KK
ISN 0031            IF(K-N) 1,2,5
ISN 0032          5 DO 12 I=1,N
ISN 0033            DO 11 J=1,N
ISN 0034            IF(ID(J)-I) 11,8,11
ISN 0035          8 DO 10 K=1,N
ISN 0036            AINV(I,K)=A(J,N+K)
ISN 0037         10 CONTINUE
ISN 0038         11 CONTINUE
ISN 0039         12 CONTINUE
ISN 0040            RETURN
ISN 0041        999 PRINT 1000
ISN 0042       1000 FORMAT(19H MATRIX IS SINGULAR)
ISN 0043            RETURN
ISN 0044            END
```

```
*OPTICNS IN EFFECT*      NAME=  MAIN,OPT=01,LINECNT=59,
*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,DECK,NOLOAD,NOMAP
*STATISTICS*     SOURCE STATEMENTS =      43 ,PROGRAM SIZE =     7844
*STATISTICS*  NO  DIAGNOSTICS GENERATED
****** END CF COMPIIATICN ******

     COMPILER STATISTICS:  ELAPSED TIME      2.500 SEC.
*STATISTICS*  NO  DIAGNCSTICS THIS STEP
```

C. The Execution Program

This program reads in the data from file Data and solves the equations using the fast algorithm in (9.5.5). The whole program has only two control cards:

```
$RUN COMPUT 7=DATA
$ENDFILE
```

Output of the program is listed in next page. The result shows that the total execution time in solving a linear system of 64 complex equations (128 unknowns) has been reduced to 2.18 seconds.

The maximum relative error occurs in the imaginary value of f_{62} where

$$\text{Max Rel Error} = \frac{1.0000000 - 0.9999946}{1.0000000} = 0.000548$$

The values of $f_1 \ldots f_{64}$ computed by solving equations Pf = F. The expected values should be either 2 +i or 0 + 0i.

F(1) =	0.0000018	0.0000007
F(2) =	0.0000026	-0.0000011
F(3) =	-0.0000016	-0.0000017
F(4) =	-0.0000025	-0.0000015
F(5) =	-0.0000028	-0.0000000
F(6) =	-0.0000026	-0.0000007
F(7) =	0.0000006	-0.0000019
F(8) =	0.0000023	-0.0000010
F(9) =	0.0000011	-0.0000023
F(10) =	0.0000016	-0.0000008
F(11) =	0.0000011	0.0000005
F(12) =	-0.0000007	0.0000017
F(13) =	-0.0000004	0.0000001
F(14) =	0.0000007	-0.0000026
F(15) =	-0.0000015	-0.0000059
F(16) =	0.0000014	-0.0000029
F(17) =	-0.0000011	0.0000010
F(18) =	0.0000009	0.0000005
F(19) =	0.0000034	-0.0000048

F(20) =	0.0000024	0.0000005
F(21) =	0.0000066	0.0000018
F(22) =	0.0000053	0.0000051
F(23) =	0.0000017	0.0000091
F(24) =	-0.0000068	0.0000029
F(25) =	-0.0000047	0.0000018
F(26) =	-0.0000051	0.0000005
F(27) =	-0.0000040	-0.0000027
F(28) =	0.0000026	-0.0000038
F(29) =	0.0000043	-0.0000016
F(30) =	0.0000047	0.0000034
F(31) =	0.0000059	0.0000015
F(32) =	-0.0000014	0.0000057
F(33) =	-0.0000007	0.0
F(34) =	1.9999952	0.9999986
F(35) =	2.0000010	0.9999999
F(36) =	0.0000010	-0.0000013
F(37) =	0.0000002	0.0000039
F(38) =	-0.0000032	0.0000026
F(39) =	-0.0000058	0.0000000
F(40) =	-0.0000038	-0.0000030
F(41) =	0.0000023	-0.0000041
F(42) =	0.0000041	-0.0000015
F(43) =	0.0000020	0.0000018
F(44) =	0.0000011	0.0000028

FORM NO. 8X
FORMALINE - MOORE BUSINESS FORMS

F(45) =	-0.0000021	0.0000086
F(46) =	1.9999895	1.0000010
F(47) =	1.9999924	0.9999982
F(48) =	-0.0000033	-0.0000073
F(49) =	-0.0000007	-0.0000010
F(50) =	1.9999971	0.9999972
F(51) =	1.9999991	0.9999965
F(52) =	0.0000038	-0.0000015
F(53) =	-0.0000001	-0.0000005
F(54) =	-0.0000014	-0.0000020
F(55) =	-0.0000005	-0.0000025
F(56) =	0.0000004	-0.0000005
F(57) =	0.0000010	0.0000018
F(58) =	0.0000008	0.0000012
F(59) =	0.0000012	0.0000002
F(60) =	-0.0000005	-0.0000001
F(61) =	-0.0000022	-0.0000019
F(62) =	1.9999981	0.9999946
F(63) =	2.0000010	0.9999955
F(64) =	0.0000049	-0.0000002

UNIVERSITY OF ALBERTA COMPUTING SERVICES (SYSTEM AR205)

USER: DKKL

```
**** ON AT  13:19.08     SAT MAY 03/75
**** OFF AT 13:19.22     SAT MAY 03/75
**** ELAPSED TIME          .233 MIN.
**** CPU TIME USED         2.18  SEC.
**** CPU STOR VMI           .516 PAGE-MIN.
**** WAIT STOR VMI          .052 PAGE-HR.
**** CARDS READ            8
**** LINES PRINTED       522
**** PAGES PRINTED        18
**** DRUM READS           48
**** RATE FACTOR          1
**** APPROX. COST OF THIS RUN IS     $1.60
```

APPENDIX IV

DIGITIZATION OF A SCATTERING FORMULA

The theoretical derivation in Chapter VII arrived at the formula (7.1.7), which is

$$u_{sc}(X') = \int_V G(X'/X)\ f_s(X)\ u_o(X)\ dV \qquad \text{(A4.1)}$$

where $u_{sc}(X')$ is the optical field scattered by a weakly scattering object of microscopic size

$f_s(X)$ is the scattering potential of the object

$u_o(X)$ is an incident field

$G(X'/X)$ is a Green's function chosen to be $-\exp(ik_o|X'-X|)/4\pi|X'-X|$.

Integration over the volume V will first be performed inside each blocklet and then be summed up blocklet by blocklet.

$$u_{sc}(X') = \sum_v \int_{V_v} G(X'/X)\ f_s(X)\ u_o(X)\ dV\ . \qquad \text{(A4.2)}$$

Putting in the far-field approximation given in (7.1.8), and (7.1.10);

$$u_{sc}(X') = \frac{-\exp(ik_o|X'|)}{4\pi|X'|} \sum_v \int_{V_v} \exp(-iK\cdot X)\ u_o(X)\ f_s(X)\ dV. \qquad \text{(A4.3)}$$

Let the vector X be resolved into two components X_v and δX.

$$X = X_v + \delta X \quad . \tag{A4.4}$$

X_v will be a vector pointing to the center of the vth blocklet, as defined in (7.2.1). δX will be a vector from the center of the vth blocklet pointing to any point X within the vth blocklet.

$$u_{sc}(X') = \frac{-\exp(ik_o|X'|)}{4\pi|X'|} \sum_v \int_{V_v} \exp[-iK\cdot(X_v+\delta X)]$$

$$u_o(X_v+\delta X)\ f_s(X_v+\delta X)\ dV \quad . \tag{A4.5}$$

The incident field $u_o(X_v+\delta X)$ will be split up into two parts.

$$u_o(X_v+\delta X) \equiv u_o(X_v)\ \frac{u_o(X_v+\delta X)}{u_o(X_v)} \equiv u_o(X_v)\ u_o(\delta X) \quad . \tag{A4.6}$$

Physical meaning of (A4.6) can be seen if, for instance, $u_o(X)$ is a plane wave propagated along the direction of Z-axis. In this case,

$$u_o(X) = \exp(ik_o z) \quad , \tag{A4.7}$$

$$u_o(X_v+\delta X) = \exp[ik_o(z_v+\delta z)] \quad , \tag{A4.8}$$

$$u_o(X_v+\delta X)/u_o(X_v) = \exp(ik_o\delta z) = u_o(\delta X) \quad . \tag{A4.9}$$

$u_o(X_v+\delta X)$ may be considered as separable into two waves: $u_o(X_v)$ acts like a carrier wave, which is actually the incident wave $u_o(X)$ measured at the center of the vth blocklet. $u_o(\delta X)$ is a modulator wave, which looks after

the phase difference within the vth blocklet. It shows that in this case $u_o(\delta X)$ is independent of v.

From (A4.5), and writing $f_s(X_v+\delta X)$ as $f_s(\delta X)$ since X_v is constant for the integral over V_v,

$$u_{sc}(X') = \frac{-\exp(ik_o|X'|)}{4\pi|X'|} \sum_v \exp(-iK\cdot X_v)\, u_o(X_v) \int_{V_v} \exp(-iK\cdot\delta X)\, u_o(\delta X)\, f_s(\delta X)\, dV. \quad \text{(A4.10)}$$

Define f_{sv} as a weighted statistical average of f_s over the vth blocklet:

$$f_{sv}(X_v) \equiv \frac{\int_{V_v} \exp(-iK\cdot\delta X) u_o(\delta X) f_s(\delta X)\, dV}{\int_{V_v} \exp(-iK\cdot\delta X) u_o(\delta X)\, dV}. \quad \text{(A4.11)}$$

In many cases, $f_s(X)$ varies very slowly within the tiny space of a blocklet. We can then assume

$$f_{sv}(X_v) \approx f_s(X_v). \quad \text{(A4.12)}$$

Then the left hand side of (A4.11) becomes $f_s(X_v)$. The numerator on the right hand side of (A4.11) is the integral in (A4.10). Let us call the denominator of (A4.11) $\alpha(K)$.

$$\alpha(K) \equiv \int_{V_v} \exp(-iK\cdot\delta X)\, u_o(\delta X)\, dV. \quad \text{(A4.13)}$$

When the incident wave u_o is a plane wave along the Z-direction,

$$\alpha(K) = \int_{V_v} \exp(-iK.\delta X)\exp(ik_o\delta z)\,dV$$

$$= \int_{-\Delta x/2}^{\Delta x/2}\int_{-\Delta y/2}^{\Delta y/2}\int_{-\Delta z/2}^{\Delta z/2} \exp(-iK_x\delta x - iK_y\delta y - iK_z\delta z + ik_o\delta z)\, d(\delta x)d(\delta y)d(\delta z)$$

$$= \frac{\sin(K_x\Delta x/2)}{K_x\Delta x/2}\cdot\frac{\sin(K_y\Delta y/2)}{K_y\Delta y/2}\cdot\frac{\sin[(K_z-k_o)\Delta z/2]}{(K_z-k_o)\Delta z/2}\cdot\Delta V$$

$$= \phi(K)\Delta V, \quad \text{where } \phi(K) \text{ is called a form factor.} \tag{A4.14}$$

This shows that $\phi(K)$ is independent of v. It is the same for all blocklets. (A4.14) also shows that when Δx, Δy, Δz approach zero, i.e. the blocklets become very small, $\phi(K)$ approaches 1, $\alpha(K)$ approaches ΔV. $\phi(K)$ is finite if $\Delta xK_x, \Delta yK_y, \Delta z(K_z-k_o)$ are not multiples of 2π.

Substituting (A4.11) through (A4.14) into (A4.10),

$$u_{sc}(X') = \frac{-\exp(ik_o|X'|)}{4\pi|X'|}\sum_v \exp(-iK\cdot X_v)u_o(X_v)f_s(X_v)\phi(K)\Delta V. \tag{A4.15}$$

Since $\phi(K)$ and ΔV are both independent of v, they can be taken out of the summation symbol.

$$u_{sc}(X') = \frac{-\exp(ik_o|X'|)\phi(K)\Delta V}{4\pi|X'|}\sum_v \exp(-iK\cdot X_v)u_o(X_v)f_s(X_v). \tag{A4.16}$$

Let

$$F(K) \equiv \frac{u_{sc}(X')\ 4\pi|X'|}{\exp(ik_o|X'|)\phi(K)\Delta V}\ , \quad K = k_o \frac{X'}{|X'|}\ ; \qquad \text{(A4.17)}$$

$$f(X_v) \equiv -u_o(X_v)\ f_s(X_v)\ . \qquad \text{(A4.18)}$$

Finally, (A4.16) becomes

$$F(K) = \sum_v \exp(-iK\cdot X_v)\ f(X_v)\ . \qquad \text{(A4.19)}$$

APPENDIX V

PROPAGATION OF LIGHT IN A QUASI-HOMOGENEOUS MEDIUM

Let the behaviour of light in a quasi-homogeneous medium be described by the Maxwell equations (A5.1) and material equations (A5.2) and (A5.3).

$$\text{curl } H - \frac{1}{c}\frac{\partial}{\partial t} D = \frac{4\pi}{c} j \,,$$

$$\text{curl } E + \frac{1}{c}\frac{\partial}{\partial t} B = 0 \,, \qquad \text{(A5.1)}$$

$$\text{div } D = 4\pi\rho \,,$$

$$D = \varepsilon E \,, \qquad \text{(A5.2)}$$

$$B = \mu H \,. \qquad \text{(A5.3)}$$

For a non-conductive and non-magnetic medium, the electric charge density ρ and the electric current density j are both zero, and the permeability μ is equal to unity. Hence

$$\text{curl } H - \frac{1}{c}\frac{\partial}{\partial t} D = 0 \,, \qquad \text{(A5.4a)}$$

$$\text{curl } E + \frac{1}{c}\frac{\partial}{\partial t} H = 0 \,, \qquad \text{(A5.4b)}$$

$$\text{div } (\varepsilon E) = 0 \,. \qquad \text{(A5.4c)}$$

Applying the operator curl to both sides of (A5.4b),

$$\text{curl curl } E + \frac{1}{c}\frac{\partial}{\partial t}\text{curl } H = 0 \ . \qquad \text{(A5.5)}$$

Putting (A5.4a) into (A5.5),

$$\text{curl curl } E + \frac{1}{c^2}\frac{\partial^2}{\partial t^2} D = 0 \ . \qquad \text{(A5.6)}$$

Substituting (A5.2) into (A5.6),

$$\text{curl curl } E + \frac{\varepsilon}{c^2}\frac{\partial^2}{\partial t^2} E = 0 \ . \qquad \text{(A5.7)}$$

Remember the mathematical relations between the vector operators curl, div, and grad (see, for example, Pipe: Mathematics for Engineers and Physicists. Chap.15, p.394, McGraw Hill 1958):

$$\text{curl curl } E = \text{grad div } E - \nabla^2 E \qquad \text{(A5.8)}$$

$$\text{div } (\varepsilon E) = \varepsilon \text{ div } E + E \cdot (\text{grad } \varepsilon) \ . \qquad \text{(A5.9)}$$

From (A5.7) and (A5.8),

$$\nabla^2 E - \text{grad div } E + \frac{\varepsilon}{c^2}\frac{\partial^2}{\partial t^2} E = 0 \ . \qquad \text{(A5.10)}$$

Taking grad of (A5.9), substituting it into (A5.10), also using (A5.4c),

$$\nabla^2 E + \frac{\varepsilon}{c^2}\frac{\partial^2}{\partial t^2} E + \text{grad } \left(\frac{E}{\varepsilon} \cdot \text{grad } \varepsilon\right) = 0 \ . \qquad \text{(A5.11)}$$

Equation (A5.11) describes, in exact mathematical terms, the optical wave E propagated in a non-homogeneous, non-conductive, non-magnetic medium. Since the material

is non-homogeneous, ε should be a function of (x,y,z). For some physical problems, ε varies very slowly in comparison with the variation of an optical field E in space. In this case, the last term in (A5.11) is negligibly small and may be dropped. To justify this simplification, let us estimate the actual values of the three terms in (A5.11). As a very rough estimate, we shall assume that E is a plane wave.

Section 7.1a gives $E = \mathrm{Re}[u \exp(-i\omega t)]$. In the case of a plane wave propagated in z-direction,

$$E = \mathrm{Re}[A \exp(-i\omega t - ik_o z)] . \qquad \text{(A5.12)}$$

Then

$$\nabla^2 E = \frac{\partial^2 E}{\partial z^2} = \mathrm{Re}[(-k_o^2) A \exp(-i\omega t - ik_o z)] = -k_o^2 E . \qquad \text{(A5.13)}$$

As we all know, $k_o = 2\pi/\lambda_o$ and λ_o is in the neighborhood of $6(10^{-7})$ meters. This shows that the first term $\nabla^2 E$ in (A5.11) is of the order $(10^{14})E$ meters^{-2}.

The second term in (A5.11) is

$$\frac{\varepsilon}{c^2}\frac{\partial^2 E}{\partial t^2} = \frac{\varepsilon}{c^2}\mathrm{Re}[(-\omega^2) A \exp(-i\omega t - ik_o z)]$$

$$= -\varepsilon E\omega^2/c^2 = -\varepsilon E/\lambda_o . \qquad \text{(A5.14)}$$

$|\varepsilon|$ is usually greater than 1 and less than 100. Then the second term is of the order of $(10^{12})E$ to $(10^{14})E$

meters^{-2}, or has a similar magnitude as the first term.

Regarding the third term grad$(\frac{E}{\varepsilon} \cdot \text{grad } \varepsilon)$ in (A5.11), if ε varies slowly in space, we can expect

$$(10^{14}) \text{ meters}^{-2} >> |\frac{1}{\varepsilon} \nabla^2 \varepsilon| \ . \qquad \text{(A5.15)}$$

Equation (A5.15) means

$$|(10^{14})E| >> |\frac{E}{\varepsilon} \text{ grad}\cdot\text{grad } \varepsilon| \simeq |\text{grad}(\frac{E}{\varepsilon} \cdot \text{grad } \varepsilon)| . \qquad \text{(A5.16)}$$

In this case, the third term is of much smaller order than the other two terms. Equation (A5.11) can be written with the third term neglected. Thus

$$\nabla^2 E + \frac{\varepsilon}{c^2} \frac{\partial^2}{\partial t^2} E = 0 \ . \qquad \text{(A5.17)}$$

Substituting (A5.14) into (A5.17) and writing $\varepsilon = (2\pi n)^2$,

$$\nabla^2 E + (2\pi n)^2 \omega^2 E/c^2 = \nabla^2 E + n^2 (2\pi/\lambda_o)^2 E$$

$$= \nabla^2 E + k_o^2 n^2 E = 0 \ . \qquad \text{(A5.18)}$$

The last equation corresponds to equation (7.1.1) if E is replaced by its scalar representation u. This equation is valid subject to the condition (A5.15). Again, if ε is replaced by $(2\pi n)^2$, we have the restrictive condition (A5.15) given by

$$|[\nabla^2 (n^2)]/n^2| << (10)^{14} \text{ meters}^{-2} \ . \qquad \text{(A5.19)}$$

APPENDIX VI

SOLUTION OF A NON-HOMOGENEOUS HELMHOLTZ PARTIAL DIFFERENTIAL EQUATION

We are going to solve the equation in (7.1.6) for u_{sc}. The equation takes the form:

$$\nabla^2 u_{sc} + k_o^2 u_{sc} = f_s u_o \qquad \text{(A6.1)}$$

or $$\nabla^2 u_{sc} = - k_o^2 u_{sc} + f_s u_o \ . \qquad \text{(A6.2)}$$

Since the scattered field is due to multiple point sources of density $f_s u_o$, contained within the volume V, we shall choose the point-source Green's function

$$G = \frac{\exp(ik_o|X' - X|)}{|X' - X|} \ . \qquad \text{(A6.3)}$$

This Green's function has two properties:

1. Differentiating (A6.3) gives

$$\frac{\partial G}{\partial |X' - X|} = G\left(ik_o - \frac{1}{|X' - X|}\right) . \qquad \text{(A6.4)}$$

2. This Green's function satisfies

$$\nabla^2 G + k_o^2 G = \delta(X' - X) \qquad \text{(A6.5)}$$

or, $$\nabla^2 G = \delta(X' - X) - k_o^2 \, G \ . \qquad \text{(A6.6)}$$

We shall make use of the second form of Green's theorem, which states

$$\int_V (G\nabla^2 u_{sc} - u_{sc}\nabla^2 G)\,dV \equiv \int_A (G\nabla u_{sc} - u_{sc}\nabla G)\cdot dA \ . \qquad (A6.7)$$

The left hand side of (A6.7), on substitution with (A6.2) and (A6.6), gives

$$\int_V [-Gk_o^2 u_{sc} + Gf_s u_o + u_{sc} k_o^2 G - u_{sc}\delta(X'-X)]\,dV$$

$$= \int_V [G\, f_s u_o]\,dV - u_{sc}(X) \ . \qquad (A6.8)$$

The right hand side of (A6.7), taking the gradients in the direction of (X'-X), becomes

$$\int_A \left(G\,\frac{\partial u_{sc}}{\partial(X'-X)} - u_{sc}\,\frac{\partial G}{\partial(X'-X)}\right) dA \ . \qquad (A6.9)$$

Substitution of (A6.4) turns (A6.9) to

$$\int_A \left(G[\frac{\partial u_{sc}}{\partial(X'-X)} - u_{sc} i k_o + \frac{u_{sc}}{|X'-X|}]\right) dA \ . \qquad (A6.10)$$

We have the radiation condition (see, for example, Butkov 1968, p.538 and p.617)

$$|X'-X|\,(\frac{\partial u_{sc}}{\partial|X'-X|} - u_{sc}\, i k_o) \to 0, \quad \text{when } |X'-X|\to\infty. \qquad (A6.11)$$

Also,

$$\frac{u_{sc}}{|X'-X|} \to 0 \ , \quad \text{as } |X'-X| \to \infty \ . \qquad (A6.12)$$

Therefore (A6.10), or the right hand side of (A6.7), disappears under far-field condition (7.1.9).

Equating the left hand side of (A6.8) to zero, we have

$$u_{sc} = \int_V G\, f_s\, u_o \, dV \qquad \text{(A6.13)}$$

which is the required solution (7.1.7) to equation (7.1.6). This solution expresses the scattered field u_{sc} in terms of the source distribution $f_s u_o$ for any point X' inside or outside the scatterer, in particular for X' in the far-field region.